Diploma in Software Development

DSE780 Application Development Project

(Capstone Project)

Alternative Human-Computer Interfaces

Project Team:

Pavel Sobolev Student ID: EDZ00005KO

Final report

11 April 2019

STUDENT DECLARATION

Student Name: Pavel Sobolev

Student IDs: EDZ00005KO

Signatures: ______________________________

Date: 11/04/2019

Table of Contents

1 EXECUTIVE SUMMARY

The primary aim of this capstone project is to research and to implement some nonstandard ways of communication between computer system and a user. Such communication can be realized through usage of relatively new and alternative interfaces. When I use words "alternative interfaces" I mean the such hardware and software which can obtain data from a person using non-standard ways (by reading user's brain data for example). These data can be interpreted as a source for controlling of computer software or hardware components connected to the computer.

Interaction between computers and humans can be realized in different ways. This project is devoted to studying and implementing relatively new and modern ways of such communication using brain-computer interface (BCI) and eye movement tracking devices.

BCI enables monitoring the user's brain activity (types of activity and levels of activity). Signals are picked up by BCI hardware and can be interpreted by software and used to control devices and other software systems. BCI can also be used to provide a user with neurofeedback which helps us understand inner mental processes and evaluate their modality and intensity.

Eye movement tracking devices use electrooculography to detect user's oculomotor activity. Output from such devices can be used to control behaviour of hardware and software. Movements of eyeballs detected by this type of devices can be mapped to appropriate movements of the screen pointer or actions on UI elements.

Possible users of such interfaces are people interested in widening their experience with the use of personal computers and making the experience more natural and more comfortable. People with certain disabilities can benefit the most from these interfaces.

Project "GazeMindDriver" is should allow people to use their natural physical abilities such as their sight or mind concentration to control computer and devices which are connected to this computer.

This project is dedicated to the development and implementation of a compound hardware and software system for creating an interface between a human and a computerized robotic system. The original idea is to control the robot's movements using an oculomotor interface and a brain-computer interface. This type of system can serve as a prototype for a professional system that can provide handicapped people (with some sorts of immobility) with the ability to control the movement of some motorized transport device using eye movements and monitoring the brain activity of the subject.

2 INTRODUCTION

2.1 Background

While studying the course "770 Robotics and Internet of Things" I developed a system which enabled interaction between a user and smart home devices utilising a brain-computer interface (the system included mobile application software and specially designated hardware). That project gave me new knowledge and some experience in the area of BCIs and their practical usage. Successful realization of the mentioned system induced further interest in this area and motivated me to go deeper and research new interfaces and their possible implementations. Using devices of traditional human-computer interfaces (such as keyboard or mouse) in combination with ETI (eyes tracking interface) and BCI in the same project is a challenge for me. This project allows to improve my personal skills in software engineering and to meet the learning outcomes of the "DSE780 Application Development Project".

Interaction between a computer and a human can be implemented in different ways. The way which is used for interaction between them is called human-computer interface (Zaphiris & Ang, 2008). Classical human-computer interfaces emerged from the principle of functioning of computing devices. They significantly rely on and depend from the capabilities and inner architecture of used computing facilities. Classical computer is provided with built-in capability to get data from outside and send data outside through its peripheral devices and inner data streams (input and output data streams). Very important to notice that classical Von Neumann's computer (which factually is a sophisticated calculator) is not originally capable to adapt itself to abilities, necessities, and peculiarities of a user (Aspray, 1990). Vice versa, a human should train and adapt himself in order to be able to use a computer. There is traditional hardware for interaction between computers and their users such as mice, trackballs, touch screens, keyboards etc. Interaction with computer by using of these devices can be called convenient in some sense but cannot be called absolutely natural.

More natural ways of interaction were always desired by wide range of computer's users. This can be possible not only by developing of hardware component of computing devices but also by implementation of modern technologies in the area of software, such as artificial intelligence, machine learning, neural networks, big data storage and processing, internet of things and smart home technology, natural language processing etc. All these factors should create new reality where computers study their users and adapt themselves to them, but not vice versa. In this reality natural interfaces will allow people to use computers without preliminary learning efforts and time consumption. Another crucial field of usage of such interfaces is help and assistance for people with physical disabilities.

Nowadays many technologies for implementing of human-computer interface exist. Let's enumerate some the most interesting and prominent of them (Kumar & Arjunan, 2016):

1. Brain-computer interfaces;
2. Muscle's electricity detection interfaces (myoelectric interfaces):
   a. Hand gesture recognition (Thalmic Labs, 2018),
   b. Eyes movement detection (electrooculography) (The Tobii Group, 2018),
   c. Mouth movement detection (leaps and/or tongue movement detection (glossokinetic potential detection)).
3. Interfaces based on technology of video and infrared emission capturing:
   a. Eyes movement detection and recognition,
   b. Head moving detection and recognition,
   c. Hands movement detection and recognition,
   d. Leaps movement detection and recognition, voiceless speech recognition.
4. Speech recognition interfaces:
   a. Microsoft Cortana, Apple Siri, Google assistant, Yandex Alisa, Amazon Alexa.

Design and creation of a computer application which uses modern and alternative ways of communicating with users can be considered as a topical task. Design and usage of such systems can improve the overall quality of user experience and make it more natural. Furthermore, there are some groups of users which can benefit from such types of software:

- people with physical disabilities who are not able to use standard HCIs easily and effectively,
- specialists whose activity demands operate computer-controlled machines and other devices in nonstandard ways (for example, in dangerous environments, or environments in which it is necessary to isolate the device from direct contact with the user),
- computer game users or educational software users, who would like to be able to control the game or the app more naturally and quickly (by gazing at some area of monitor and/or by changing the intensity of their mental processes),
- people interested in personal mental and psychological self-improvement.

2.2 Goal

The goal of this capstone project is to design a desktop software for providing users with eyes movements detection and tracking interface (with addition of brain-computer interface). The capstone project focusses mainly on studying ways of implementing such interfaces in modern software systems. The following problems are planned to be explored:

- Hardware components, software components (operating systems, drivers, protocols, services) for creation and implementation of a robotized movable system.
- Application programming interfaces and software development kits and technologies for implementation of eyes movements detection and tracking interface (ETI),

- Application programming interfaces and software development kits and technologies for implementation of brain-computer interface (BCI),
- Development of software for utilising of APIs and SDKs of ETI and BCI for controlling of the robotized movable system.

2.3 Scope

The software being developed should implement following functions:

1. eyes movements detection and tracking interface for Microsoft Windows (in the form of dynamically linked libraries),
2. brain-computer interface for Microsoft Windows (in the form of dynamically linked libraries),
3. wireless network intercommunication between PC equipped with an eye tracker and BCI device and remote computer, which realizes functionality of robotized movable system (MS Windows, in the form of dynamically linked libraries).
4. combining of ETI, BCI for driving a robot being implemented in this project (MS Windows, in the form of standard binary executable file).

Overall idea of design of software and hardware system can be represented by the following diagram.

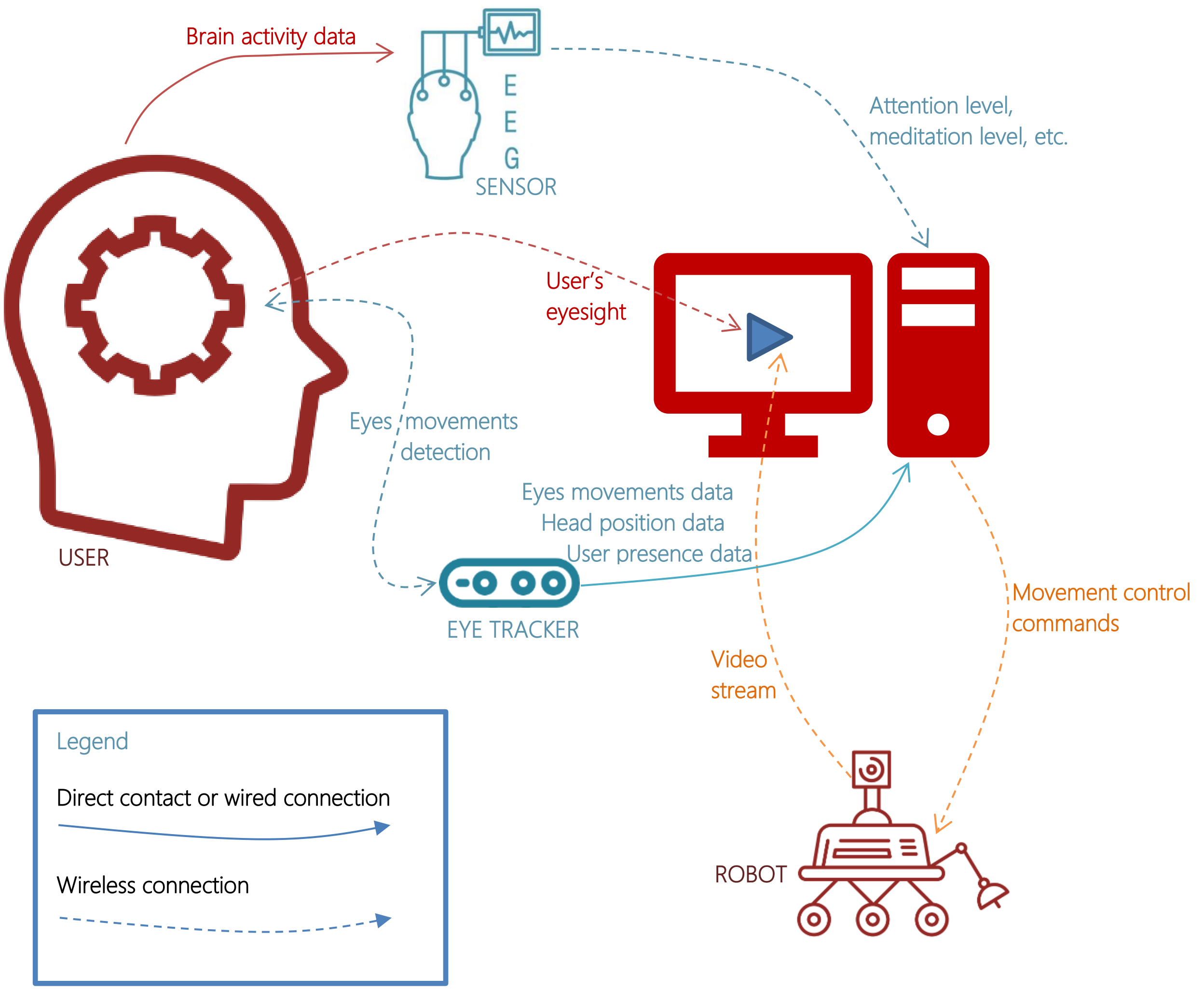


Figure 1: Description of overall system architecture to be implemented

This diagram depicts conceptual approach to the formulated tasks. It is planned to be incarnated in the form of system of interconnected software and hardware components. Preliminary scheme correspondence between parts of provided system's architecture and software components to be implemented is presented in table 1.

Table 1: Description of planned implementation of required functionality

Implemented functionality	Software component	Form of implementation
Network connectivity.	1. HTTP connectivity component. 2. SSH connectivity component.	1. DLL 2. DLL
Creation and sending of robot's commands.	1. Component for SSH command implementation 2. Component for GPIO command implementation 3. Component for motion command implementation	1. DLL 2. DLL 3. DLL
Interaction with gaze input device.	Component for capturing of gaze input (data and events) received from gaze input device.	DLL
Interaction with EEG headset.	Component for capturing of data received from the EEG headset.	DLL
Interaction with video data from the robot.	Component which should provide visual representation of video stream from the camera for main app of the project.	DLL
Interaction with graphical user interface	The app which uses mentioned components and combines their functionality with its UI to achieve main goals of the software being designed.	EXE

3 PROJECT PLANNING AND EXECUTION

3.1 Project Members

Project Team members	Name	Student ID	Email	Phone
	Pavel Sobolev	EDZ00005KO	pavelsobolev@outlook.com	+64211433235

3.2 Methodology

Software development lifecycle is a process which controls both overall and concrete activities within the process of creation of software product. According to studied sources (Langer, 2016; Pressman, 2001) taking of decision about model of SDLC to be used in some project is not easy decision and it should take in consideration a lot of different aspects. But some concrete considerations can be listed according to the mentioned above sources and existing definitions and descriptions of modern models of SDLS being used in the software engineering. Concerning "K-OGL" project set of questions were compiled and on the base of the prevailing answers final SDLC model was chosen.

I plan to use iterative incremental model (Bitlner & Spence, 2006) of software development life cycle. This will allow me to develop the system by leading it through a sequence of several phases where each new phase adds some new feature, functionality or somehow improves quality characteristics of the software being developed. This type of SDLC also allows constant reviewing, testing, discussions of the project state with project's coordinator and presenting the project during the classes. One of the most thoroughly detailed and precise implementations of iterative incremental model is called IBM Rational Unified Process (Barnes, 2007; Boggs & Boggs, 2002) is based on object-oriented modelling and analysis and uses standardised Unified Modelling Language (UML) (Larman, 2004). Studying and using this methodology can contribute to overall improvement of efficiency of the software development process.

Table 2: analysis of SDLC models *for this capstone project* (Bitlner & Spence, 2006; Langer, 2016)

Characteristic of the project / Does model correspond this characteristic?	Waterfall	V-model	Iterative incremental
Main part of the requirements is known and clear but is going to evolve during the process of development.	✗	✗	✓
Not all the requirements were defined preliminary.	✗	✗	✓
Requirements are to be changeable.	✗	✓	✓
There is a need to implement some requirements on early stages of the project (especially it concerns connectivity of devices and basic functionality for interaction with hardware).	✗	✓	✓
Project demands development of overall system architecture in the area which is new for the developer (eye movements tracing, BCI, assembly and programming of a robot, etc.).	✗	✗	✓

There are some unknown issues of software development in this project (programming of wireless network connectivity, programming of new types of devices, programming of access to cloud data storage and local isolated data storage).	✗	✗	✓
Tools of development relatedly new to the developer.	✗	✓	✓
There is not team working on the project. Changing of roles is not applicable.	✓	✓	✗
It is possible to get some additional training for the project developer during the process of development.	✓	✓	✓
Customers are going to be involved in the process of development (in the person of the project's coordinator).	✗	✓	✓
Possible users can evaluate current condition of the project (in the person of the project's supervisor and developer).	✗	✓	✓
Consumers are planned to participate in each phase of project SDLC.	✗	✗	✓
Consumers (in the person of the project's coordinator) can observe the process of development.	✗	✗	✓
This project isn't middle or large scaled project.	✓	✗	✗
The project doesn't extend existing projects.	✓	✓	✗
This project is expected to be exploited for some period in the future.	✗	✓	✓
Ready program is expected to be changed after finishing of the project.	✗	✓	✓
The project has strict time limitation.	✓	✓	✗
Created components of the projects can be reusable.	✗	✗	✓
There are enough resources for the development of the project (time, development tools).	✓	✓	✗
Overall score	6/20	12/20	15/20

So, after comparison of results of undertaken analysis it is can be said that iterative incremental model of SDLC is the most suitable for this capstone project. Further discussion of chosen DSLC follows below in the next chapter of this report.

3.3 Deliverables

Table 3: Planned deliverables of the capstone project

Deliverable	Description	Contribution to the project

The robot (hardware and software system to be controlled by eyes movements detection and tracking and the state of user's brain activity).	Movable metal platform equipped with computer, motorized wheels and video camera.	15%
Binary dynamic libraries for MS Windows	Implementation of brain-computer interface functionality.	5%
Binary dynamic libraries for MS Windows	Implementation of eyes movements detection and tracking interface functionality.	10%
Binary dynamic libraries for MS Windows, Apache Web-Server scripts	Implementation of video streaming from the robot (sending and receiving).	10%
Binary dynamic libraries for MS Windows	Implementation of wireless network communication between desktop application being developed and the robot (creation, sending, receiving and running commands).	10%
Binary dynamic libraries for MS Windows	Implementation of ways of combination and interpretation of data which are received from the eyes movements detection and tracking device and brain activity reading device for formation of robot's movements commands.	10%
Windows desktop application	Implementation of major functionality of the system being developed.	20%
Test cases	Unit tests, functional tests, performance tests.	10%
Report	Detailed description of the project.	5%
Presentation	Demonstration of the results of the project.	5%

3.4 Tasks and Milestones

Table 4: Major tasks and milestones of the capstone project

Task	Date due
Analysis of the capstone demands and its learning outcomes, discussion of the project's topic with the coordinator.	22 November, 2018
Creation and editing of the project proposal, handing-in of the project proposal.	22 November 2018
Overall system planning. Risks estimation. Elicitation of requirements. Modelling of the software. Designing of the system architecture.	29 November 2018
Analysis of hardware and software platforms for brain-computer interface. Implementation of binary executables for BCI in MS Windows OS.	6 December 2018
Analysis of hardware and software platforms for eyes movements detection and tracking interface. Implementation of binary executables for BCI in MS Windows OS.	13 December 2018
Development of desktop software for implementation of BCI and ETI for practical use. Implementation of web-camera streaming from Raspberry Pi computer to Windows desktop application.	10 January 2019

Implementation of combining of data from BCI and ETI with data from video stream.	17 January 2019
Assembly of the of the robot. Implementation of data transfer from desktop application to Raspberry Pi computer for using it to control a GPIO. Testing of created software. Fixing bugs.	24 January 2019
Implementation of commands for controlling of the robot. Commands should provide direct control over LAN connection, GPIO and SSH.	31 January 2019
Design of the app's overall architecture. Choice of design patterns. Implementation of classes which realize the app's functionality. Implementation of unit tests for app's classes being created.	21 February 2019
Development of desktop software for implementation of ETI (and BCI) for practical use (converting eye tracing data to executable robot's commands).	28 February 2019
Development of desktop software for implementation of ETI (and BCI) for practical use (converting eye tracing data to executable robot's commands).	7 March 2019
Testing of created software. Fixing bugs.	14 March 2019
Overall documenting of development process and results of previously undertaken activities. Report writing.	21 March 2019
Overall testing of the entire system. Creation of test cases.	28 March 2019
Finishing of the final report. Providing of the report for peer review.	4 April 2019
Final complete project presentation.	11 April 2019

3.5 Resource Requirements and Costs

Resources of the project include software components and additional, specific hardware components in addition to the personal computers which are used in the development process. The hardware components needed are a brain-computer interface device and an eyes movements detection and tracking device. Total cost of these hardware components are based on their current price in Amazon.com: BCI-device ("NeuroSky MindWave Mobile 2") (NeuroSky Inc., 2018) is NZD160, and "Tobii eye tracker" device (The Tobii Group, 2018) is NZD230. The faculty already has both of the devices (the second was bought specially for this project in December of 2018). Another crucial part of the project is the robot (movable and programmatically controlled device) which is going to be implemented on the basis of Raspberry PI computer (under Linux operating system). Movability is going to be implemented by using of DC motors with attached wheels. Main element of the interaction with the remote computer which should control the robot is the online video stream being sent from the video camera installed in the frontal part of the robot. All the mentioned hardware components are already possessed by the faculty.

Needed software components mainly will be the hardware's APIs and SDKs which are provided for free by the hardware vendors (NeuroSky, Tobii, Raspberry). Other APIs, repositories, libraries, frameworks, and operating systems are all open source or already paid and possessed for by the faculty (i.e. operating systems, cloud services and so on).

Only free of charge and free of copyright restrictions assets (like graphical resources and other multimedia resources) will be used in the project.

3.6 Risk Analysis Checklist

3.6.1 Generic Risk Checklist

The following risk checklist is a generic model and is used to give an overall (non-specific) picture of the project's risk factors. It can also be used to compare the relative risks to the organisation of several different projects (table 5).

Table 5: The project's generic risks

	Low (Yes)				RISK			(No) High		
	0	1	2	3	4	5	6	7	8	9
Inherent Risks										
Project Objectives										
Is the project small?							▨			
Is the project of minor importance to the business?			▨							
Is the project functionally straightforward?	▨									
Are several parties able to define the requirements?	▨									
Is the subject area well documented?			▨							
Are preceding projects well documented?			▨							
User Organisation										
Does the project maintain existing user procedures?							▨			
Is other organisational change unlikely during the project?	▨									
Are the users grouped in one location?	▨									
Technology										
Is tried hardware being used?				▨						
Is tried software being used?				▨						
Can custom programming be avoided?										▨
Is the project technically straightforward?							▨			
Is the quality of existing data good?		▨								
Acquired Risks										
Scope and Approach										
Is the project scope well defined and agreed?		▨								
Is the project approach well defined and agreed?				▨						
Project Organisation										
Are people's roles clearly defined?	▨									
Are users committed to the project?	▨									
Are staff able to commit sufficient time to the project?	▨									
Are the required skills available?			▨							
Does backup exist for all members of the project?	▨									
Are political and personal relationships good?	▨									
Is the project independent of third parties?					▨					
Can a small group achieve the design?	▨									
Can a "Big Bang" implementation be avoided?	▨									
Experience, Training and Support										
Does the IT team know the technology?					▨					
Do the users know the technology?								▨		
Is the technology well supported?					▨					

3.6.2 Specific Project Risks

Risk management is a methodology which allows developers to elicit possible problems that can happen during the process of development and, thus, be prepared in the case of their emerging. Three parts of risk management can be called: assessment of possible risks, mitigation of risks which happened or precautions for prevention or mitigation, and risk costs evaluation (Stoneburner, Goguen, & Feringa, 2002). On the base of analysis of sources (Arnuphaptrairong, 2011; Wallace & Keil, 2004) all software project risks can be categorized by so called risk dimension, which defines the perspective of view for determining source of possible emerging risks. According to mentioned authors 6 risk dimensions can be elicited: risks coming from users or customers; risks connected with software requirements; risks coming from project complexity; risks from project planning and control; risks coming from the project team; risks coming from organizational environment; risks coming from the used equipment and software. According to this approach four possible groups of risks for this project were detected (table 6).

Table 6: Specific capstone project risks

#	Possible risk	Possible effect	Precautions for prevention
1	Hardware and electrical equipment risks		
1.1	Issues of the device of eyes movements detection and tracking.	Deadline of project can be broken. Not suitable or adequate data of eyes movements being provided.	Thorough following of the device instructions. The app should analyse received data in order to detect erroneous, incomplete, inconsistent or defected information. The app must handle errors and exceptions of used device during the process of its execution.
1.2	Issues of the device of brain-computer interface.	Deadline of project can be broken. Not suitable or adequate data of brain activity are provided.	
1.3	Issues of automation device (the robot).	Deadline of project can be broken. Impossible to implement demanded functionality and provide adequate reactions to commands being sent.	Thorough following of the device instructions. Thorough and careful assembly of robot. Usage of correct electrical equipment. Following the basic rules of electrical safety. Only approved (according technical specifications) and tested data should be sent to the controlled device.

1.4	Computer failures (first of all, data storage system failures).	Possibility to lose software projects files, failure of projects due dates.	Using of software for control versioning; usage of cloud storages.
1.5	Failures of hardware and equipment suppliers (or delivery services) because some components were bought online.	Due dated can be broken.	Choice of reliable suppliers of used hardware (such as Amazon.com) or nearly located suppliers (in New Zealand or in Australia).
2	Problems with requirements		
2.1	Requirements don not correspond to actual abilities and functions of used hardware.	Project's functionality does not correspond to the required.	Thorough studying of hardware of the project. Discussion of requirements with project's coordinator before starting the project and during its implementation according to chosen used SDLC methodology.
3	Problems with overall architecture of the system.		
3.1	Not correct or not adequate choice of the system nodes (devices, protocols) due to lack of experience in the area of the project.	Failure of due date of the project because of possible	Discussion of architecture components with the coordinator during all the SDLC of the project. Short public presentations for groupmates.
4	Software related problems including testing		
4.1	Lack of experience in the area of eyes movements detection and tracking programming, brain-computer interface programming. Not enough experience of programming and handling of streams of empirical data.	Not correct conclusions about data which are being received during sessions of working with the device. Not correct algorithms applied to the data being received.	Studying of manuals of manufacturer of eyes movements detection and tracking device and BCI-device about data which is provided by the concrete device and their sense in the app being developed. Studying of materials about eye movements tracking and EEG and sense of its data (brain waves data (attention, relaxation), gaze events (saccades, fixations), etc.).
4.2	Lack of experience of programming of robotized systems.	Choice of not correct or not enough adequate services which are not compatible with other components of the system being designed.	Thorough study of existing services and their functionality, prices and limitations. Discussion of robot's components with the project's coordinator during all stages of the SDLC.

3.7 Quality Assurance Process

The purpose of this section is to describe planned activities within quality assurance process (Swanson, 2016; Vance, 2013) of this capstone project. These activities must guaranty timely delivery of the system which complies the project's requirements (section 5.2). Overall quality assurance process verification and validation is going to be provided by continuous weekly consultations with project's coordinator, public presentations of current state of the project and providing trial copies of software for groupmates for their testing and feedback. Hereafter I enlist principal quality characteristics of the project and how I'm going to achieve them during the development process (table 7).

Table 7: Description of quality assurance process for this capstone project

Quality Characteristic.	Meaning of the characteristic concerning this project.	Verification activities *or* Activities which allow to achieve needed quality of characteristic.
Functionality: suitability	Functions of the app are adequate to specified tasks.	Functional testing. Presentation of the project for the project coordinator and groupmates.
Functionality: interoperability	The system should provide interaction between the desktop app and the robot.	Integration testing.
Reliability: fault tolerance	The system should continue its working (without runtime errors) in the case when network connection faults happen, or hardware (eyes tracker, EEG headset, the robot) function improperly.	Periodical ad-hoc testing. Back-end testing. Failover tests.
Reliability: recoverability	The app should be capable to restore state of the robot after appearing of connection. Also, the app should be able to recover data which was used at the moment of faults (caused by hardware components).	Recovery testing.
Usability: operability	The app must allow the user to operate it during all the period of its functioning. It means that the user should have ability to send commands to the robot by all available means at any time.	Acceptance testing (by groupmates and the project coordinator).

Efficiency: time behavior	Functions of the app should be executed during specified time in responsive manner (under concrete stated conditions).	Measuring of the app's resources during its work. Profiling and performance testing.
Efficiency: resource utilization	The app's usage of system resources should not exceed minimal requirements (section 5.2).	Performance testing.
Quality of software architecture		
Loose coupling (Hunt & Tomas, 2000)		Creation of independent classes within the app's structure where each class can interchange data with other class by sending messages and events. Peer review of the code.
Cohesion (Pressman, 2001)		Thorough analysis of system's required functions. Separation of different types of functions and responsibilities to different modules and classes of the modules. Peer review of the code.
Reusability (Hunt & Tomas, 2000)		Creation of extendable architecture on the basis of interfaces and abstract classes (Liskov substitution principle). Here I mean that the app's code should be available for adding new types of interaction and for other types of input devices. Peer review of the code.
Quality of software modules implementation		
Source code quality		Unit tests. Peer review of the code. Usage of special software tools.
Quality of implementation of needed functionality of each unit of the app.		Unit tests.

3.8 Project Work Plan

Table 8: The project's Gantt chart

	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9	Week 10	Week 11	Week 12	Week 13	Week 14	Week 15	Week 16
Analysis of the capstone demands and its learning outcomes, discussion of the project's topic with the coordinator.	▨															
Creation and editing of the project proposal, handing-in of the project proposal.		▨														

Overall system planning. Risks estimation. Elicitation of requirements. Modelling of the software. Designing of the system architecture.			▨													
Analysis of hardware and software platforms for brain-computer interface. Implementation of binary executables for BCI in MS Windows OS.				▨												
Analysis of hardware and software platforms for eyes movements detection and tracking interface. Implementation of binary executables for BCI in MS Windows OS.					▨											
Development of desktop software for implementation of BCI and ETI for practical use. Implementation of web-camera streaming from Raspberry Pi computer to Windows desktop application.						▨										
Implementation of combining of data from BCI and ETI with data from video stream.							▨									
Assembly of the of the robot. Implementation of data transfer from desktop application to Raspberry Pi computer for using it to control a GPIO. Testing of created software. Fixing bugs.								▨								
Implementation of commands for controlling of the robot. Commands should provide direct control over LAN connection, GPIO and SSH.									▨							
Design of the app's overall architecture. Choice of design patterns. Implementation of classes which realize the app's functionality. Implementation of unit tests for app's classes being created.										▨						
Development of desktop software for implementation of ETI (and BCI) for practical use (converting eye tracing data to executable robot's commands).											▨	▨				
Development of desktop software for implementation of ETI (and BCI) for practical use (converting eye tracing data to executable robot's commands).													▨			
Testing of created software. Fixing bugs.														▨		
Overall documenting of development process and results of previously undertaken activities. Report writing.															▨	
Overall testing of the entire system. Creation of test cases.												▨	▨	▨	▨	
Finishing of the final report. Providing of the report for peer review.																▨

3.9 Reflection on preliminary activities

Choice of the topic for the project was made on the basis of my personal interests. During last decades I found that I more and more interested and intrigued by new ways of communication between users and computing devices. From my perspective of view modern state of this area is far from perfection. I mean that modern computers have power to work efficiently and fast as never before, but not still capable to adopt itself to the needs and peculiarities of their users. User experience and efficiency of his work still dramatically depend on limitations of modern human-computer interfaces. We create text documents by typing letters with keyboards (this way cannot be called nor fast, nor efficient) and point to objects on the screen by moving a mouse on the surface of the desktop. But these can be done more efficiently by direct dictation and using gaze's direction detection correspondingly. New technologies for such kind of interactions emerged during last decades. This project is an attempt to bring new abilities to the area of standard desktop applications by making these applications more natural in the sense of accepting some physical data of users such as eye movement directions and mind activity (derived from EEG characteristics). I believe that such projects can promote the idea that we need some new ways for controlling our devices and we actually already have instruments for implementation of these new ways. Two types of user's data were chosen for this project (direction of eyes and brain activity). First of all, eyes movements detection and tracking should be mentioned. This technology allows to catch and understand where the point of user's attention at each concrete moment of time is. This technology is absolute new for me which makes the project more interesting in the sense of learning new APIs and SDKs. Experience of implementation of this technology in my project can give me possibility to delve into technologies of programming of streams of data, to search for proper ways of realization of gaze tracking. Another challenge is design and assembly of a robot. This task lays a bit aside from software engineering, but it is impossible to deny that hardware is a material basis of any kinds of software. As a software developer I want to know and understand how to reach my goals in the area of programming through constructing appropriate hardware components. This can widen my professional experience and let me feel more confident in such industry fields where skills of relatively low-level programming are in demand. The idea of this project emerged and evolved in discussions with the project's coordinator. My original idea was to utilize user's gaze data for implementation of some training application which could be used to provide its users with set of exercises promoting their ability to use eyesight to control graphical user interface. But this approach wouldn't provide me with ability to crate really advanced project in the sense of used architectural, technological and other aspects because of relatively narrow problem field. After testing of new equipment (Tobii eye tracker) and consultations with the coordinator we've decided to design a system which involves not only software but also hardware parts which could interact by the means of eyes movements detection and brain wave data registration.

4 SOFTWARE DEVELOPMENT LIFE CYCLE

4.1 Justification of choice

According to previous analysis (see section 3.2) incremental iterative methodology was chosen for this project (Bitlner & Spence, 2006). This means that development should go through repetitive stages and each new repetition is devoted to widening and improving of the existing functionality. Also new functionality can be introduced, or existing functionality can be removed in the case if requirements changed since the time of last revamping of the project. But incremental iterative SLDC is the common approach which should have concrete materialization in the project. Considering that object-oriented technology was chosen for implementation of the project I decided to find and use namely object-oriented implementation of incremental iterative SDLC. In general words, this implementation should support and somehow describe and document phases of incremental iterative process. Documentation and description of phases must be based on the usage of key conceptions of object-oriented paradigm. This idea means that object-oriented analysis and design should be used during the SDLC.

Object-oriented analysis and design are documented and accompanied with artifacts created with Unified Modelling Language (UML) (Booch, 1991). Artifact is defined as a document, diagram and other descriptive entity which accompanies and explains each concrete phase of the Rational Unified Process.

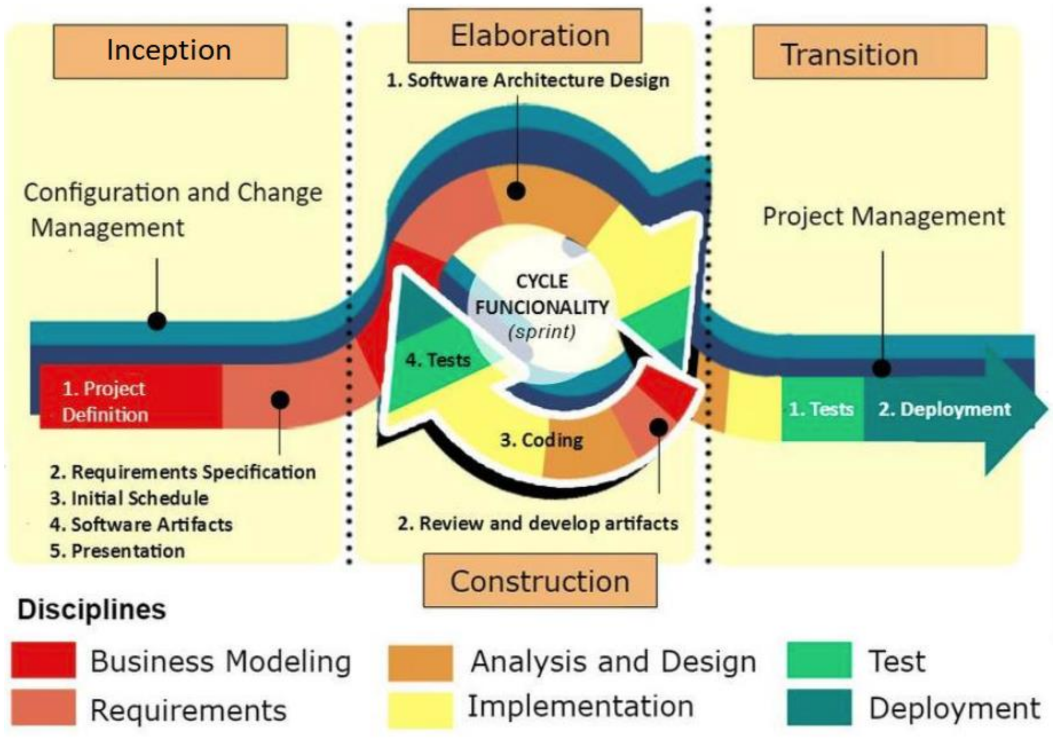


Figure 2: graphical description of Rational Unified Process (©ResearchGate, 2019)

Concrete form of implementation of chosen incremental iterative methodology was created by Rational (subdivision of IBM) is called Rational Unified Process (RUP) (Barnes, 2007). The common idea of this methodology is to provide concrete framework which can be adapted to the needs of some project. This adaptation uses object-oriented models during each phase of the project implementation which can be considered as a great benefit of RUP for this project because it is implemented on the base of usage object-oriented paradigm.

Authors of this lifecycle model (Rational company , 2001) define the process as a sequence of activities which should be decomposed into several stages (or phases) which are inception, elaboration, construction and transition. The process relies on some basic principles: incremental iterative software development, management of requirements, using of component-based architectures, usage of visual modelling, verification of software quality, control of changes during the process of development. Following diagram (figure 2 (ResearchGate, 2019))© shows structure of the process.Let's analyze some strong and weak sides of using of RUP in my project (Boggs & Boggs, 2002). From used sources I could conclude that RUP has some significant benefits for my project.

1. <u>RUP is based on strong object-oriented approach.</u>
   This allows me to use holistic approach to all the stages of the software development. From the beginning to the stage of testing and other final activities pure object-oriented technologies can be used. By contrast, frequently used functional approach (series of IDEF notations and processes) not always provides such seamless way of transition from modelling to software coding in the sense of usage of homogenous concepts. IDEF artifacts not always directly support and correspond to the ideas of object-oriented programming.
2. <u>RUP uses UML.</u>
   This fact means the same as in the previous item. The crucial part among all other types of UML diagrams is a class diagram. This diagram emerges after some preceding stages of analysis and has direct mapping to software code. Some CASE programs (Computer Aided Software Engineering) (such Rational Rose (Boggs & Boggs, 2002)) is capable to generate code of object-oriented programming languages on the base of UML diagrams (this is called forward engineering; nevertheless, this technique is not going to be used in this project). Object-oriented nature of UML allows to effectively use it as a source of ideas for software design.
3. <u>RUP provides logical decomposition of the scope through the usage of interconnected models.</u>
   This fact means that there is a sanitized and well-developed set of steps which can lead the project to the needed result (if participants of the development process comply the rules and conventions). Usually following models are used: use case model, analysis model, design model, deployment model (often each model has prefix word "business" which is used when complex enterprise system is under construction).
4. <u>RUP artifacts can be used by any participant of the project as a source of information because of its graphical representation.</u>
   UML notations have relatively simple rules and structure which can be comprehended even by people with non-engineering or non-mathematical background. This eases communication between team members. If to say about my project, usage of well diagrammed models can show logical structure of project.

Meanwhile, RUP methodology has some known *limitations* (Admiraal, 2019). Concerning this capstone project, usage of Rational Unified Process may look as some sort of overkill. Normally it is recommended to use such methodology for large scale projects which involve many participants with diverse roles. But anyway, this is a learning project and experience of using of such methodology can be beneficial for learning outcomes and my personal learning goals. Another drawback of RUP is that participants of the project should somehow be familiar with the ideas of UML. But most of UML diagrams have relatively simple notation rules and uses intuitively and easily comprehensible graphical blocks and arrows for their connection. So, this kind of limitation should not be an unsurmountable obstacle for the participants of the project.

Following chapters describes actives which were undertaken during the process of analysis, design and development of software and hardware system. According to the overall aims of the project some steps of RUP were omitted (for example, business modelling) because the project is not designated for immediate business use and is not connected to some company or enterprise.

During the process of the app development I used the following sequence of my analytical activities (which is recommended by used sources (Boggs & Boggs, 2002; Booch, 1991)): goals analysis →

system use case design →

description of the flow of events →

classes design and data design →

sequences/activities analysis and design →

coding →

testing.

This chain of actions can be repeated as demanded through the process of incremental SLDC being used in order to refine result of the previous phase.

4.2 Implementation of chosen SDLC in the project

This section is devoted to description of how I implemented chosen SDLC in my project. The whole process of the app's design was decomposed into several parts according to initial project's plan (table 4, section 3.4) and used SDLC framework. Each part is an iteration which contained of standard phases: inception, elaboration, construction, and transition.

4.2.1 Description of activities during the process of development (1st iteration)

4.2.1.1 Inception

At the beginning stage of development, the primary goals were formulated:

1. Development of application which should implement gaze input interface and brain-computer interface. In general case, this app should somehow use data from such devices as eye tracker and EEG headset.
2. Learning of technologies of eye movement tracking and find device for implementation of such feature in the app. A device should have ability to detect eyes movements without a need to limit movability of a user (in other word this device must catch natural flow of human gazing). Also, price for a device should be affordable.
3. Choosing of technologies for realizing of the goals.

Users of the system were defined as people who can benefit from using of mentioned interfaces (because of problems with health or due to their personal reasons).

Simplified use-case diagram shows the initial vision of the system (figure 3).

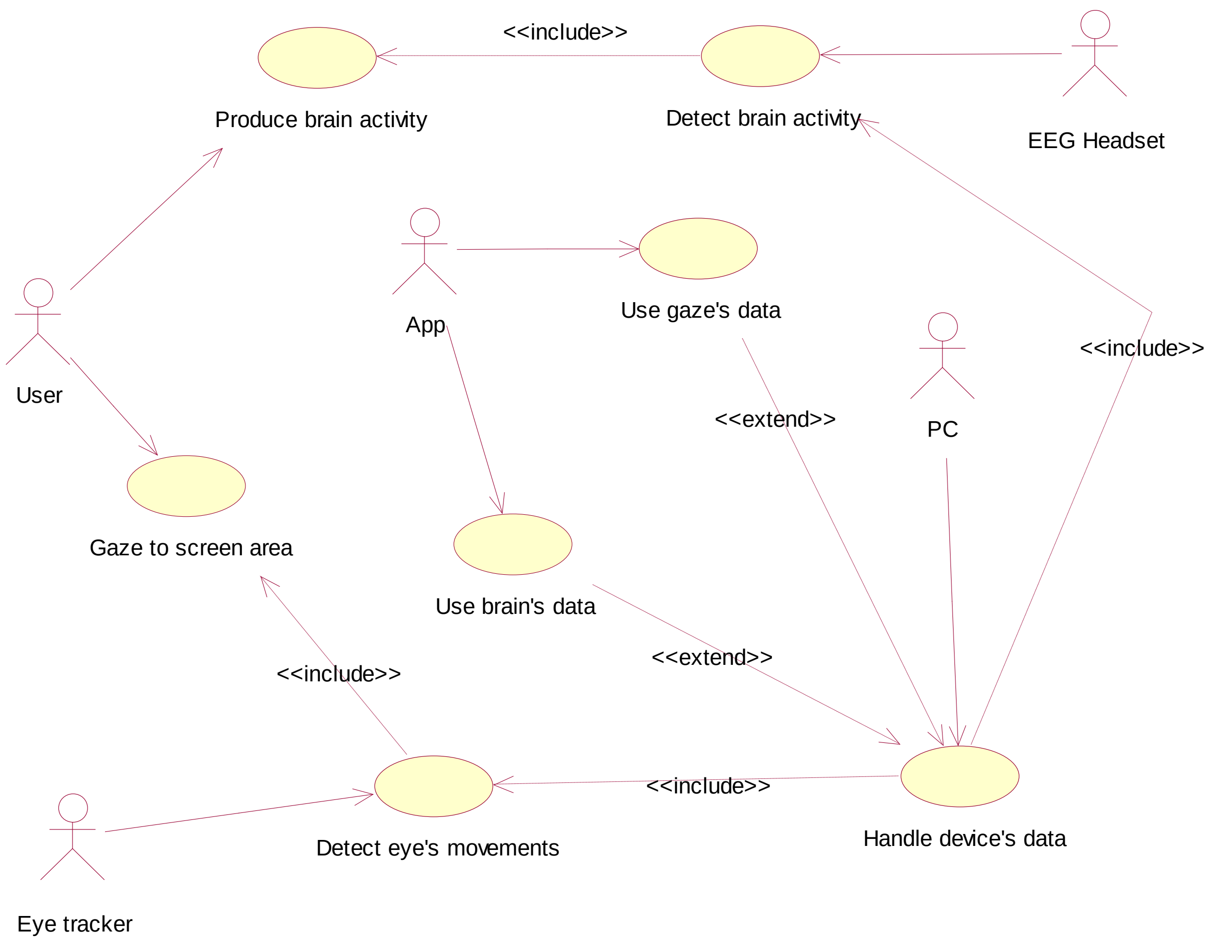


Figure 3: Initial simplified use-case model

4.2.1.2 Elaboration and construction

In this phase "Tobii eye tracker 4C" was ordered from Amazon and received during one week after placing of order. This allowed to create first preliminary version of application and understand overall abilities of the equipment which is available for the project.

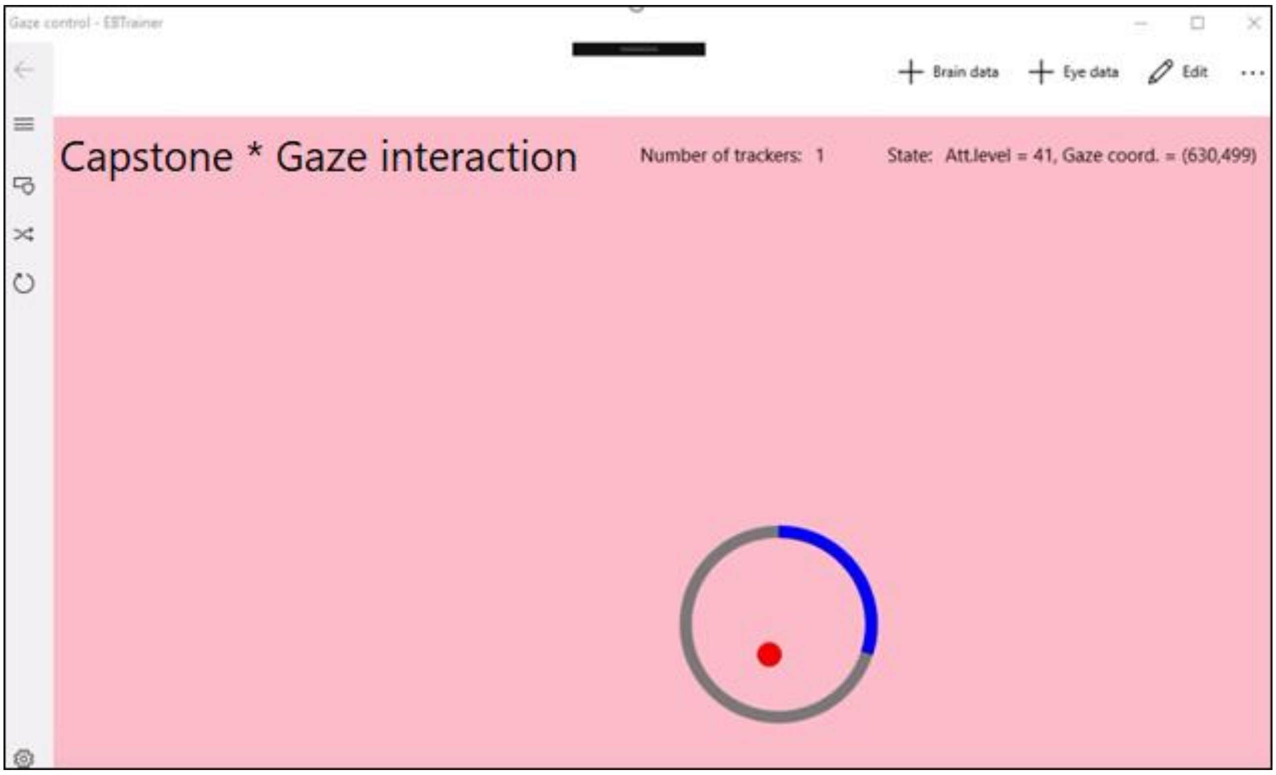
Initial plan for the system's architecture was as follows. The application should consist of one executable file which contains following classes:

1. A class for interaction with eye tracker,
2. A class for interaction with EEG headset,
3. A class for creation visual scenes (probably computer game) which can utilize data from previous classes.
4. A class for storing data of the app.

Technologies and basic software for development were chosen (justification of the choice of technologies is given later in the section devoted to implementation of the project):

1. Microsoft .NET Framework (version 4.7),
2. Microsoft Visual Studio 2017 Community Edition,
3. Microsoft Visual C# programming language (version 7),
4. Universal Windows Platform (UWP) API and XAML for implementation of UI,
5. Microsoft Gaze API (Windows.Devices.Input.Preview).

Some of technologies were new for me (4 and 5 above). According to initial use case model preliminary version of application was created to try chosen technologies and understand abilities of hardware.



Capstone * Gaze interaction

Number of trackers: 1

State: Att.level = 41, Gaze coord. = (630,499)

Figure 4: Initial version of the application (UI)

This app was capable to catch gaze point (small red circle) and EEG headset's data (level of the user's level of attention is shown in the label in the right upper corner of the window). The idea of the app was to provide a user with simple game. The user must consciously direct and hold his gaze inside the big circle during exact period of time (until blue counter fill the outer part of big circle). The higher the level of the user's brain activity the bigger the target circle.

Implementation of this application allowed me to understand functionality of the equipment and new technologies of development (UWP and preview version of Microsoft Gaze API). The main conclusion was that main problem of this system may come from the intensive stream of data from two devices. This meant the multithreading architecture should to be implemented. Otherwise, UI of the app would become unresponsive and application's usability would be quite low. First version of application implemented independent thread for reading of data from EEG headset.

This application was publicly presented and was discussed with groupmates. The overall feedback was generally positive. The main concern was said by the project coordinator who noticed that this application doesn't provide really useful functionality and doesn't show all the potential abilities of the used devices. Because of that circumstance I and project coordinator decided to improve the project functionality and change requirements. So, it's can be said that functionality of the system was to be incremented according to standard terminology of SDLC. Also, this preliminary version of the app was not intended for any kind of deployment. Since that moment next iteration of SDLC started.

4.2.2 Description of activities during the process of development (2nd iteration)

4.2.2.1 Inception

Final decision of improved version of requirements were made on the base of newly formulated global task of the project. The new task was to create a prototype of a system which could control a remote movable robotized device by utilizing data received by special equipment from user's eyes movements and user's brain activity.

Following changes and additions were made to initial list of requirements:

1. Development of application capable to control the movements of a robot by transformation of the user's eyes movements data to motion control commands (left, right, forward, etc.). Feedback from the robot should be received from its video camera which send online video stream which show real situation in front of the robot's camera.
2. Learning of technologies and equipment for design of a robot.
3. Choosing of hardware components for assembly of a robot.
4. Learning of software components and technologies for implementing sending and receiving of video data from the camera over local network connection.

On the base of new goals new version of use case model was designed (figure 5). It improved and widened my understanding of system's architecture and requirements to its functionality.

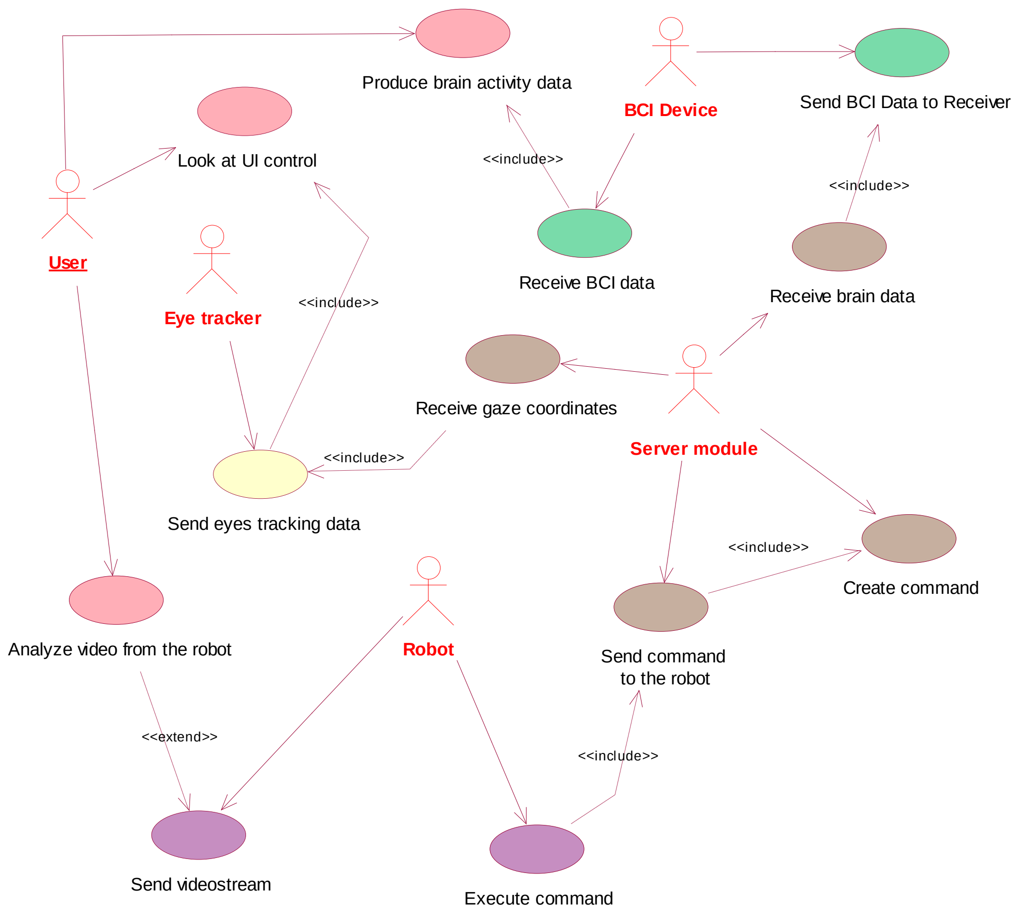


Figure 5: Second version of use case model of the capstone project

4.2.2.2 Elaboration and construction

New element of the system is the robot which delivers new tasks to overall implementation of the system. The problem of providing responsive system with affordable time frames of latency appeared. Experiments with the used equipment and software showed that latency in this system is not constant and depends from main extraneous factor which is the state of throughput channel of local network connection. Another crucial component which affects the resulting efficiency is way of how rendering of video stream is implemented in Windows application.

Initial choice for implementation of user interface of desktop app was made to the favour of the newest generation of Windows built-in API which is called Universal Windows Platform (UWP). When the robot was assembled then the latest version of the app was revamped and rendering of video from robot's camera was added to the previous version of the app. This was made just before Christmas break. During EDENZ Christmas break Microsoft released automatically installed update of Window 10 operating system. The result of this update showed extremely negative impact on my application (it is obvious that this update was intended to somehow "improve" features of UWP API being used in my app). Actual effect of this update changed efficiency and speed of rendering of video which was received from the robot's camera. Video in the window of my app became extremely flickering and slow, latency increased at least one order of magnitude comparing to initial (this latency became easily detected by eyesight). This was unacceptable. Searching for possible ways to fix this problem showed that Microsoft was not willing to improve the situation quickly. Because of that a had to rejected usage of newest and actually unstable Universal Windows Platform API and skipped to much more stable (because of its wide usage and age) Windows Forms API. Windows Forms API appeared together with .NET Framework as its part for realization of graphical user interface. Implementation of video rendering in WF allowed to decrease level of delay to the acceptable level (this means that average time of delay time did not exceeded 1 second).

Because of replaced API of UI, I had to change API for gaze input. Initially I used Microsoft Gaze API (Windows.Devices.Input.Preview) which provided rather wide spectre of functions (gaze detection, fixations detection, gaze's dwellings detection and so on). Another benefit of this library that it is capable to work with eyes tracker of any manufacturer. Concerning my project, this library has the major significant drawback. Namely, it supports only applications of UWP (which was rejected because of inacceptable latency time) and this means that the library can be used only in Windows 10 operating systems. Because of that I had to replace Microsoft Gaze API to Tobii Core SDK (provided by the manufacturer of eyes tracker being used). This SDK has wider range of supported versions of Windows and has several implementations for different development platform (it can be used from apps written in C/C++ and .NET languages). Also, this SDK provides wider range of events and data from eye tracker being used. Obvious limitation of the SDK is that it should be used only with devices of this concrete manufacturer (Tobii).

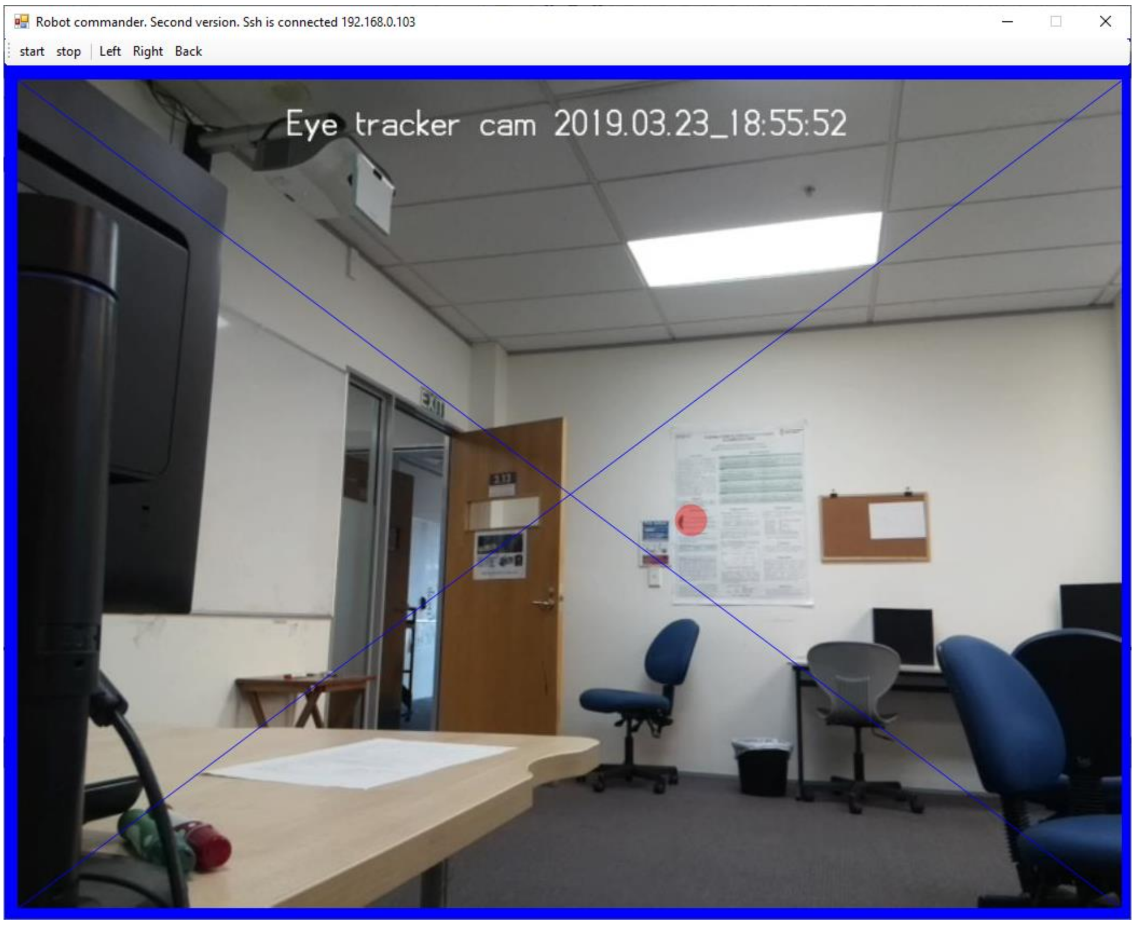

Robot commander. Second version. Ssh is connected 192.168.0.103
start
stop
Left
Right
Back
Eye tracker cam 2019.03.23_18:55:52

Figure 6: UI design of the app of the 2nd iteration

Main part of the window is occupied by picture retrieved from the robot's camera. Figure 2 shows the window of the second major version of the app. Red circle depicts point where the user's gaze was directed. Also, trial graphical output on the surface of video stream was created (blue diagonal lines). This was maid just to prove that this is possible, and to make sure that rendering doesn't negatively affects the overall efficiency (here I mean delay time of graphical output). Also, basic motion commands are represented in the app's toolbar.

New version of the app had following functions:

1. Local network connectivity with the robot;
2. Simple implementation of SSH interaction;
3. Implementation of sending video stream from the robot to PC (server side);
4. Implementation of receiving video stream from the robot;
5. Implementation of receiving data and events from eye tracker;
6. Implementation of combination of data from eyes tracker and video stream;
7. Basic implementation of robot's commands which were sent by pressing buttons of the app's toolbar.

So, implemented functionality of second version of the app closely approached desired final functionality besides the ability of transformation of gaze movements on the surface of video picture from the robot's camera to motion commands of the robot.

Initial testing activities were undertaken. Firstly, this new version was presented for groupmates and the project coordinator. Main idea of this phase was to understand how to receive video stream from the robot's camera and how to combine video data and gaze stream data. Also, new APIs were introduced to the project. This leaded to revamping of the code from the previous version of the app. Also new classes were introduced, and modular structure of the app was realized (table 1).

I created some unit tests for checking the most crucial parts of the app. Namely, components which establish internet and local network connections were tested. They have principal importance because they are needed by all other components of the app for sending data to the robot and receiving data from the robot. Also, my attention was directed to achieving stable timing of the video stream and the eyes' data stream. The main goal was to somehow guarantee acceptable level of latency. The main problem emerged from unstable quality of connection over network provided in EDENZ. Sometimes the network throughput allowed to achieve rather smooth video rendering, but sometimes picture became quite slow. Anyway, appropriate solution to the problem was partially implemented by using of multithreading.

Functionality of the app's second version was not complete and was needed to be incremented in order fully satisfy the requirements. Furthermore, the app was given to some groupmates for code review. Some valuable feedback was received. In particular, I was pointed out that hardcoded initial values for fields of classes in many modules were used. This circumstance demanded rethinking of how to implement work with constant data in the app (for example, this type data represented IP addresses and port numbers,

numbers for GPIO pins of Raspberry PI, names of SSH commands and so on). This problem was solved by introducing the app's local data storage. This problem was solved by introducing the app's local data storage in the 3rd iteration of development process.

Physical architecture of the app's components (2nd iteration) is depicted by figure 7 (UML component diagram). *Major functionality is implemented in 6 DLL's and one EXE file of desktop client-side application.* Also, server-side component was created. It functions on Raspberry PI computer which operates the robot.

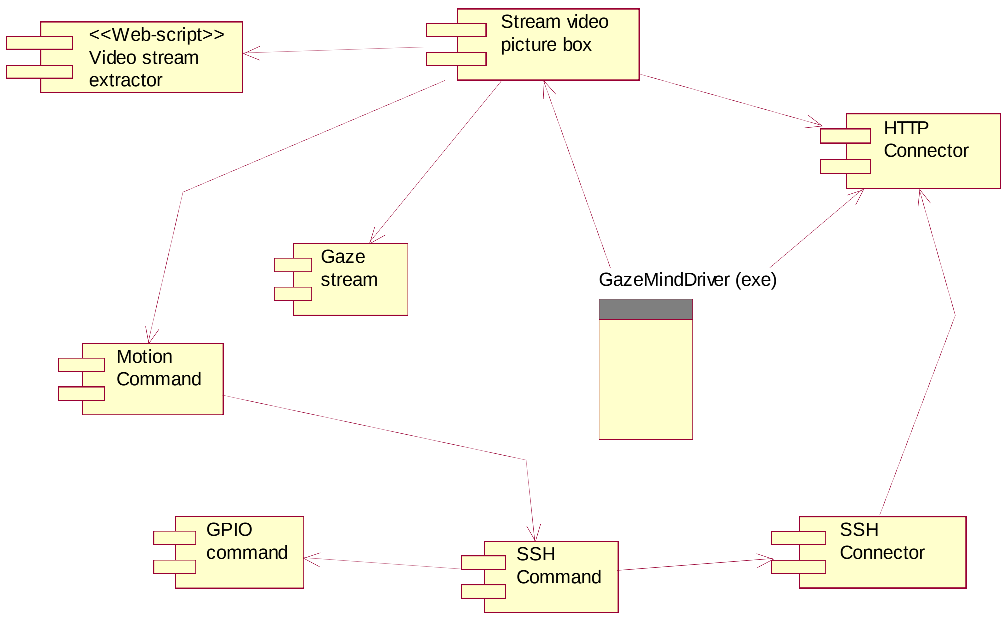


Figure 7: Component diagram for the app of 2nd iteration

4.2.2.3 Deployment

Deployment of this phase of the project was realized by assembling of the robot and connecting needed devices with PC being used. Figure 8 depicts physical structure of the project (UML deployment diagram).

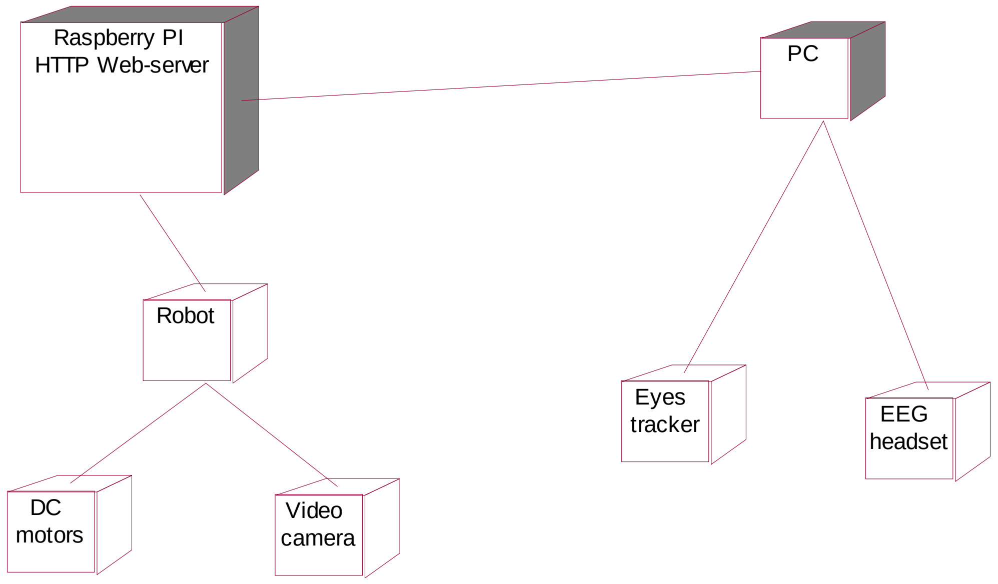


Figure 8: Deployment diagram for the app of 2nd iteration

Because of the app's incompleteness and found issues (concerning the app's source code), new iteration of lifecycle was started. Third phase of the development is described in the further sections of the report in details (because this phase was final and realized almost complete needed functionality).

4.3 Reflection on SDLC

Before starting the project, it was not clear how to select suitable SDLC because at the initial point requirements were not very strict and hardware parts were not chosen. When situation with devices to be used was resolved it became clear that incremental iterative SDLC could be used in this project (reason for such conclusion is in the table 2).

Even after receiving of equipment for eye movement detection and assembly of the robot there was some uncertainty about requirements, and they were changed several times which in general corresponds to chosen SDLC. I had to learn available functionality of the devices of hardware part of the system in order to understand how to map them to structure of the software being designed.

New iterations were needed not only because of changes in requirements but also because of issues emerged from basic software (Windows OS) when I had to skip from one API to another. This started chain of consequences which eventually demanded to create absolutely new implementation for video streaming and gaze input (see 4.2.2.2). Mentioned obstacles (4.2.2.2) didn't allow easily create the system which can be universally used with any type of eyes trackers. But on the other hand, this situation allowed to create the app which is not strictly dependent from one concrete version of Windows operating system. I had to create such the app's architecture which can be easily expanded by adding new class for new type of eye tracker. The only demand in this case is that this new class should comply and implement specially created interface (section 7.2.4).

5 SYSTEM ANALISYS AND REQUIREMENTS

5.1 Object-oriented analysis of the system

5.1.1 Applicability of object-oriented approach to this project

Object-oriented approach to the software design and development is widespread nowadays. This project significantly uses rules and conventions of this methodology to achieve the result which was formulated in the executive summary. According to designed overall architecture (figure 1) it is necessary to model programmatically a system which contains a set of initially independent objects.

These objects are the user, the personal computer and its software, data providers (eyes movement detector and electro-encephalographic detector) and a movable robot. In programmatical model being designed I should somehow connect these objects in order to achieve the main goal. Namely it is needed to force the robot execute commands which corresponds to movements of the user's eyes and activity of the user's brain. I believe that object-oriented approach can give some significant benefits over some other approaches. Mentioned architecture could be also implemented by the means of pure procedural approach (using relatively low-level languages like assembly language or C or their combination). But object-oriented methodology provides some advanced abilities in the sense of ways of modelling of the reality being programmed.

Table 9: Comparison of the project's applicable implementation approaches for needed key characteristics of future software (preliminary estimation)

Degree of applicability in object-oriented approach (OOA) (Booch, 1991)	Degree of applicability in procedure-oriented approach (POA) (Yemul, 2015)	The most preferable approach for the project
It should be possible to model used notion or object as one independent entity which can be referenced as a whole.		
This a built-in feature of OOA. This can be realized by using of classes and other derived data structures. So, it is possible to start programming from the concept to realization (top-down direction) and save time for other activities of SDLC.	Theoretically it can be achieved by constructing the custom data structure which eventually simply repeat built-in conception of a class in OOA. In this case realization should precede the modeling itself (down-top direction). For this project this approach can be too time-consuming undertaking.	OOA is more preferable because allows direct creation of needed modeling abstractions using specially predefined features of some OO programming language. In POA I would spend more time on creation ant testing such models for their correctness and adequate implementation.
Objects being modeled should interchange data by sending messages to each other.		
This can be achieved by using concept of message in OOA.	This idea can be roughly implemented by invocation of	OOA is preferable because it provides more natural way

This allows to deliver packets of data from one component of the app to another. Usage of such conception naturally corresponds to the ways of interaction between parts of the architecture being modeled.	procedures with some set of suitable arguments. But anyway, this type of implementation does not completely correspond the needed functionality. The problem is that a message handling demands clear defining of initializing part. This leads to necessity of using of callback mechanisms which system-dependent (different operating systems implement this idea in different ways). Also, implementation of continuous loops of awaiting are needed. Clear defining of a sender of a message could be difficult because such conception is not supported.	of real-time data interchange with ability to clearly define a sender and a receiver. Also, this approach could allow to create cross-platform code which doesn't depend on peculiarities of some concrete operating system.
Objects should produce events in the case when their important characteristics or states were changed. This behavior is needed for real-time reaction of the system on ongoing changes.		
Events allow to notify interested participants about fact of happening of something significant for the event's receiver. OOA languages and technologies directly support this conception. Events can be used to implement such behavior as reaction of the system when level of brain activity in some aspects (attention, for example) descends below some allowed limit.	Along with messages, events are not directly supported by procedural languages. But can be implemented by specific mechanisms which provided by modern operating systems. Anyway, this demands additional programming and usage of such problematic constructions as continuous loops and callbacks.	OOA is preferable because has native implementation of conception of event which can make structure and code of the app more concise and maintainable. POA vice versa leads to creation of cumbersome and large volume of code with not clear structure (because of the callbacks which should be inevitably used).
There is a need to implement data hiding. In some cases, important data of some object should not be reached (even for reading) from other objects of the app.		
OOA languages and technologies provide such features through usage of special decorators which cab control level of access to the parts of the classes and structs.	POA is implemented by languages which mainly doesn't have specifically designed system of control of access level. Technically this can be done by hiding data inside declarations of the functions of procedures, but this makes harder to access them from within the module where they are defined.	Concerning my project, data encapsulation is needed. This can be effectively implemented in OOA but not in POA. So, OOA is preferable in this case because provides my project with built-in ability to achieve encapsulation on the level of a programming language syntax structures.
Data sharing between different objects (for accessing global stable states of some characteristics of the system as whole).		

Sharing of data is necessary feature for the project. This circumstance follows from the fact that there are some common sources (such as network connection, IP address, system thread pool and so on) which can be used by all components of the application. Meanwhile, this common data should be realized as objects (which are capable to send notifications, receive messages and so on). This also can be done within this approach.	POA languages allow to create and define shared data by placing their declarations in any part of a module outside bodies of functions, procedures or data structures. Lack of control of access level affects this ability negatively. Shared data (as an entity) cannot define ways of its own initialization and utilization. Impossible do restrict ways of redefining of their values.	Again, OOA is preferable. OOA not only allows to effectively share data, but also do this with some prudent restrictions which can be implemented inside this shared data sources (if they are implemented as independent standalone entities). This will allow to promote transparency of the code (good readability, maintainability). Furthermore, this can help make behavior of the app more predictable and can ease testing activities (for example, unit testing).
Data abstraction should be realized in order to achieve the goals of designing of universal data structures which are suitable for extension and adaptation to behavior of different types of similar devices. This means that the app needs to use the one version of a method which can accept data of similar data types which can have their own version of common behavior.		
It would be great to follow some categorical demands of some theoretical principles in this project. According to my experience many of them especially useful. I mean segregation of interfaces and Liskov substitution principle. They are can be implemented in OO paradigm by utilizing some relatively simple ideas of this interfaces and abstract classes. They allow to use abstract definitions instead of concrete data types which promote reusability and maintainability of the code.	Data abstractions is not typical for procedural programming.	OOA is preferable due to lack of data abstraction implementation in POA.

As can be seen from the undertaken analysis, object-oriented approach can be the choice for the project implementation. According to present practice such projects (which involves hardware control) are mainly done in C or similar procedure-oriented languages (because they not so resource consuming as object-oriented or functional languages) (Clements, 2006). But recent development of computing devices made it possible to bring OOP to this area of software development. Nowadays we can easily and without significant loss of efficiency create such systems by using such languages and technologies like Java, C#

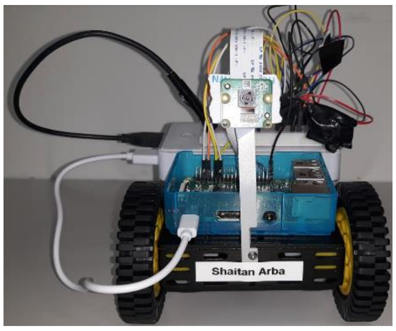
or VB.NET in .NET Framework, or C++ and its high-level frameworks (such as Qt, MFC, STL or Boost) and so on. Object-oriented approach of software design can be and should be applied on all stages of used SDLC.

5.1.2 Glossary

The glossary provides definitions of terms to be used during the process of modelling of the system and forms preliminary documentation for RUP (Rational company , 2001). According to RUP, this is an artifact which can be used to introduce specific terminology of the scope which is being modelled, for system analysts, developers, testers and other participants and stakeholders.

Table 10: Glossary of the system being designed

#	Term	Short definition / explanation
1	User	This is a person who is capable to use software being designed (healthy eye sight is needed in order to use the software; usage of the software is not recommended for people with some deceases (like epilepsy)) (The Tobii Group, 2018)).
2	Gaze	This is coordinated motion of eyes, neck and head in the process of act of visual perception (Leigh, 2015).
3	Eyes' saccades	This is a rapid and sharp movements of both eyes to the same direction between two moments of fixation (Leigh, 2015).
4	Eyes' fixation	Fixation is an act of gazing when eyes directed to some single location. Human eye sight can be roughly defined as a sequence of alternating saccades and fixations (Leigh, 2015).
5	Eyes tracking	This is the technology which consists of specially designed hardware and software and allows detect moments of eyes movements, saccades, fixations. Also, presence or absence of the user can be detected.
6	Eyes tracker	Eye tracker is a hardware part of eyes tracking technology. This class of devices is presented by set of sensors, actuators and other parts for technical implementation of eyes tracking.
7	Eyes data stream	This is a set of consecutive bytes of data which is sent by some eye tracking device to operating system accepting points (COM ports, USB ports and so on). This data should contain somehow encoded information about direction of the user's gaze and information of significant events during the gaze act (fixations, saccades, user's coming and leaving).
8	BCI	Brain-computer interface (BCI) is a technology which allows to obtain, process and use data of the user's brain activity. BCI relies on data from EEG sensors attached to the skin of the user's skull.
9	EEG	Abbreviation EEG stands for "electroencephalography". This is the method of exploring of a human brain activity. EEG produces set of data on the base of detection of so-called brain waves (NeuroSky Inc., 2018).
10	Brain wave	This is persistent pattern of electrical activity of the nerve system of the brain which can be interpreted as a description of some inner brain process (NewroSky Inc., 2018).
11	EEG Headset	This is an instrumental hardware part of BCI. Usually nowadays this is a mobile version of electroencephalographer which is capable to send data over wired (usually USB) or wireless (Bluetooth, Wi-Fi) connection to a computer (NewroSky Inc., 2018).

12	Brain data stream	This is a set of consecutive bytes of data which is sent by some BCI device to operating system accepting points (COM port or USB port or a network port and so on). This data should contain somehow encoded information about direction of the user's brain activity and contains its characteristics (in appropriate absolute or relational measurement units (usually units for measurement of electrical characteristics)).
13	Attention level	In this project brain activity is measured by two cumulative calculatable characteristics. One is attention level. This characteristic is calculated by BCI software on the base of raw data of the user's brain activity and shows how active is his mind during the period of measurement. This level is represented by integer value within interval (0;100]. The lower the number, the more user's mind is directed to outside (this regularity can be used for controlling the movements of the robot).
14	Relaxation (meditation level)	This is also the integer within interval (0;100] and shows how passive the user's mind. If level of relaxation is high, then user's ability to accept new information or to study is not good.
15	Robot	In this project the robot is the metal platform equipped with a portable mini-computer Raspberry PI, video and photo camera (one device), electrical breadboard, controllers and connectors, two DC motors (with attached wheels), one passive wheel and portable battery. The wheels allow to move the platform along some even surface. Other descriptive details of the robot in this project are given in appropriate section of the report (6.1.1). 
16	DC motor	Direct-current motor is a rotational mechanism driven by electivity and controlled by commands which are sent through special inner mechanisms of Raspberry PI computer. Motors are used in the project in order to provide movability of the robot.
17	Video stream	In this project this is a set of consecutive bytes produced by connected camera and sent over local network channel to the software. This data stream contains video data which shows current situation around robot during operations being done by the user.
18	GPIO	General purpose input output. This is hardware and software interface for interconnection between Raspberry PI computer's specially designed PINs and external hardware.
19	SSH	Secure shell which is instrument of remote access to Linux command line interface.

Other phases of software development process use these terms.

5.1.3 Goals of the project

Concerning this project, it should be said that is has not only objective purposes (like creation of hardware and software system for realizing of non-standard human-computer interface) but also personal educational purposes. And both are interconnected and influence each other. RUP provides special extension of use case diagram which is called "Goal diagram" (or "Business goal diagram"). It contains graphical block of special stereotype which is called "aim" (or "target"). This element depicts one relatively atomic and non-decomposable goal which is pursued by business (if it is used for business modelling) or goal to be achieved by creation of software being designed. I use this diagram to concisely describe aims of the project in visual form. According to executive summary (section 1) the following aims of the system can be listed (aims are connected by "dependency" arrow ("Uses a") and directed from dependent aims which can be achieved through achievement of other aims).

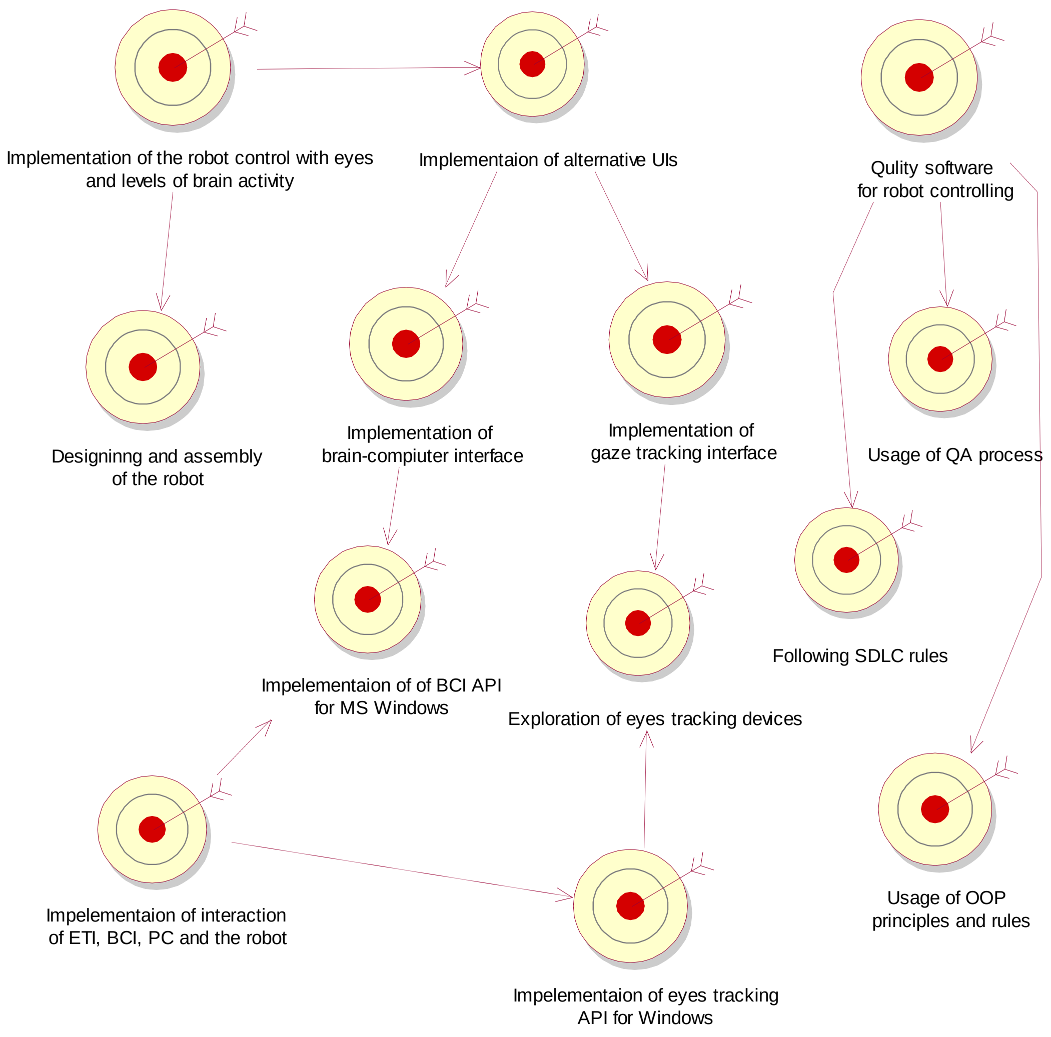

Implementation of the robot control with eyes and levels of brain activity
Implementaion of alternative UIs
Qulity software for robot controlling
Designinng and assembly of the robot
Implementation of brain-compiuter interface
Implementation of gaze tracking interface
Usage of QA process
Following SDLC rules
Impelementaion of of BCI API for MS Windows
Exploration of eyes tracking devices
Impelementaion of interaction of ETI, BCI, PC and the robot
Usage of OOP principles and rules
Impelementaion of eyes tracking API for Windows

Figure 9: Diagram of the goals of this capstone project

5.1.4 System use cases

At this stage of development, it is important to understand functionality of the app being created and who and what hardware and software components are involved during the app. Result of this analysis can be depicted visually by UML case diagrams. According to used incremental iterative conception, I had several versions of this type of diagrams, because I widened functionality several time. New functionality and structural elements (actors and use cases) emerged when each new phase began (due to necessity to realize such mandatory things like system's modularity, principle of single responsibility of each class, upgraded level of abstraction (in order to somehow universalize ways of creation and dispatching command from the user), using of architectural and design patterns). Also, new functionality emerged because I was gaining better vision and understanding of the tasks of the project and had conversations and discussions with project's coordinator and groupmates. Figure 5 (section 4.2.2.1) shows my initial understanding of future system functionality at the beginning of the project, whereas figure 10 depicts last use case model which really affected the final outcome.

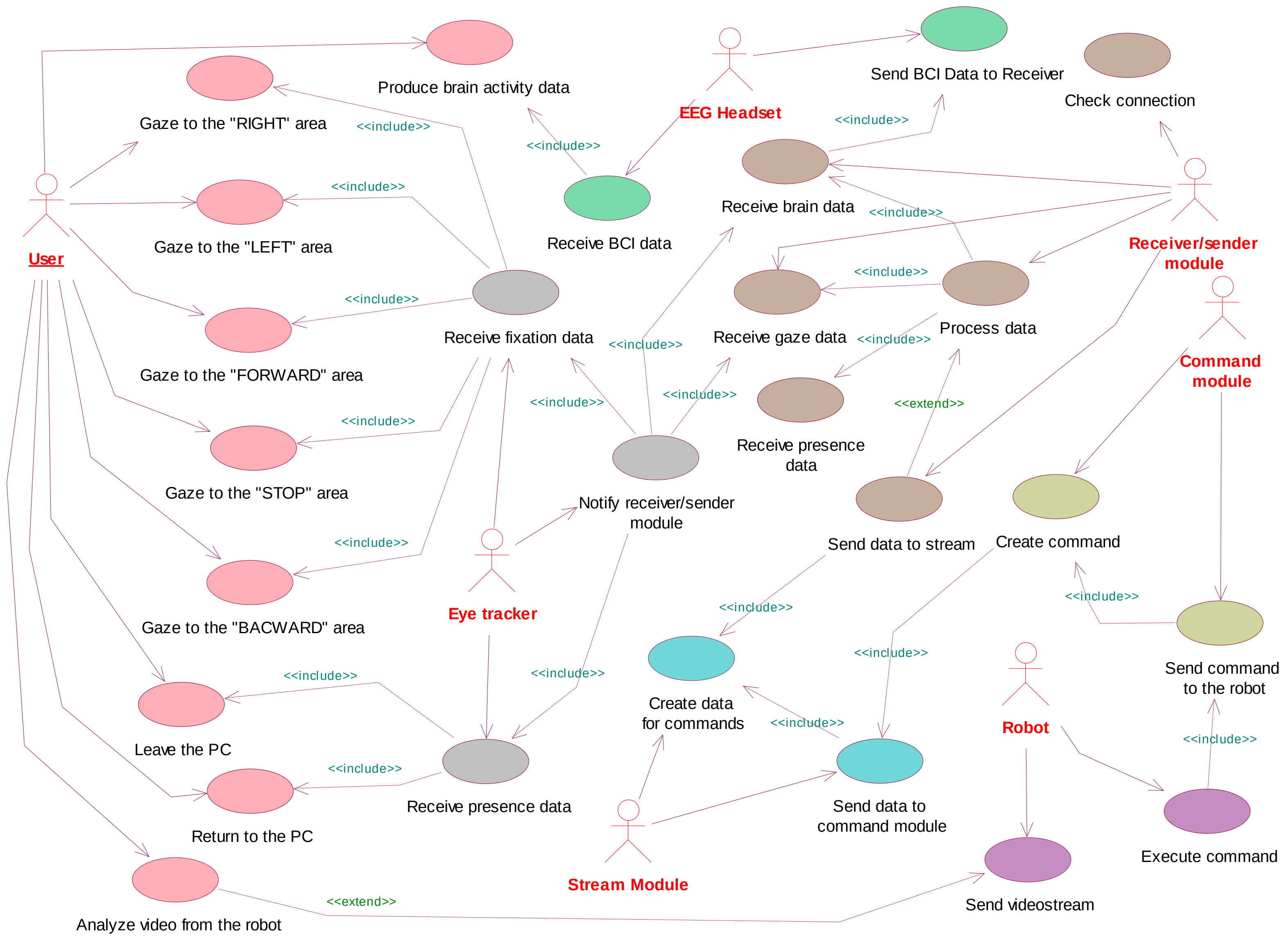


Figure 10: Final use case diagram of the capstone project

5.1.5 Flow of events

Flow of events is a document which describes on of the activities within object-oriented analysis (Boggs & Boggs, 2002). Flow of events describes what the system will do. Or, if to say more specifically, the flow of events documents the flow of logic through the use case. This description has such substantial benefit that it is implementation-independent. It is imposable to document all details of the all use cases in the text of this report. Further there will be some of the most important uses cases documented. Also, this documentation can be helpful for testing creation and execution because it describes expected system's response (and expected appropriate order of actions of this response).

Table 11: Flow of events of eyes tracking detection and usage

Description		
This use case allows a user to control remote robot by gazing to some specific area of the app's window.		
Preconditions (PE)		
PE1	Eyes tracker is properly installed and configured in the operating system being used.	
PE2	The eyes tracker is connected to a PC, turned on and is functioning.	
PE3	Drivers and other system-specific appropriate software of eyes tracker is running.	
PE4	The PC being used has Internet connection.	
PE5	The PC being used has local network connection.	
PE6	The app should be launched.	
PE7	The robot should be turned on.	
PE8	The robot's onboard computer has internet connection.	
PE9	The robot's onboard computer has connection to local network.	
PE10	The robot has appropriate hardware configuration (according to specification 6.1.1).	
PE11	The robot's onboard computer has correctly configured appropriate service software (see 6.1.2).	
Primary flow of events (PRE)		
#	User action	System response
PRE1	User allows detection of eyes movements in the app by using its command bar.	1. The app shows capturing video stream from the camera of the robot. 2. System shows specially UI element (UI_1) which capable to receive gaze input.
PRE2	The user directs his eyes to the UI_1.	1. The app goes to the state of receiving and sending events of gaze input to the robot. 2. The app shows set of UI controls (UI_2) which allows the user to give commands to robot (start, move to some specific direction or stop).
PRE3	The user directs his eyes to the UI_2.	1. The app process data of gazing. 2. The app constructs and sends a command to the robot. 3. The robot receives the command which was sent by the app. 4. The robot executes received command.
Secondary flow of events (SRE)		

SRE1	User disallows detection of eyes movements in the app by using its command bar.	1. The app stops showing capturing video stream from the camera of the robot. 2. The app hides UI_1 and UI_2. 3. The app sends command to stop the robot. 4. The robot stops (in the case if it's moving).
Flow of errors (EE)		
#	Error	System reaction
EE1	Internet connection is lost during the app is functioning	1. The app informs the user about the issue by popup window. 2. The app stops video streaming from the robot. 2. The app sends stopping command to the robot. 3. The app asks the user to stop user manually in the case if it didn't respond and didn't stop (because of inability to send command due to absence of connection channel).
EE2	Local network connection is lost during the app is functioning.	
Postconditions (PCE)		
PCE1	The app is stopped.	
PCE2	The robot is not moving.	
PCE3	The robot is turned off.	

Table 12: flow of events of eyes tracking detection and usage

Description		
This use case allows a user to control remote robot by gazing to some specific area of the app's window.		
Preconditions (PB)		
PB1	Eyes tracker is properly installed and configured in the operating system being used.	
PB2	The eyes tracker is connected to a PC, turned on and is functioning.	
PB3	Drivers and other system-specific appropriate software of eyes tracker is running.	
PB4	The PC being used has Internet connection.	
PB5	The PC being used has local network connection.	
PB6	The app should be launched.	
PB7	The robot should be turned on.	
PB8	The robot's onboard computer has internet connection.	
PB9	The robot's onboard computer has connection to local network.	
PB10	The robot has appropriate hardware configuration (according to specification 6.1.1).	
PB11	The robot's onboard computer has correctly configured appropriate service software (see 6.1.2).	
Primary flow of events (PRB)		
#	User action	System response
PRB1	User allows detection of eyes movements in the app by using its command bar.	1. The app shows capturing video stream from the camera of the robot. 2. System shows specially UI element (UI_1) which capable to receive gaze input.
PRB2	The user directs his eyes to the UI_1.	1. The app goes to the state of receiving and sending events of gaze input to the robot.

		2. The app shows set of UI controls (UI_2) which allows the user to give commands to robot (start, move to some specific direction or stop).
PRB3	The user directs his eyes to the UI_2.	1. The app process data of gazing. 2. The app constructs and sends a command to the robot. 3. The robot receives the command which was sent by the app. 4. The robot executes received command.
Secondary flow of events (SB)		
SB1	User disallows detection of eyes movements in the app by using its command bar.	1. The app stops showing capturing video stream from the camera of the robot. 2. The app hides UI_1 and UI_2. 3. The app sends command to stop the robot. 4. The robot stops (in the case if it's moving).
Flow of errors (EB)		
#	Error	System reaction
EB1	Internet connection is lost during the app is functioning	1. The app informs the user about the issue by popup window. 2. The app stops video streaming from the robot. 2. The app sends stopping command to the robot. 3. The app asks the user to stop user manually in the case if it didn't respond and didn't stop (because of inability to send command due to absence of connection channel).
EB2	Local network connection is lost during the app is functioning.	
Postconditions (PCB)		
PCB1	The app is stopped.	
PCB2	The robot is not moving.	
PCB3	The robot is turned off.	

5.1.6 The project architecture

The app architecture defines overall further activities during the development process, logical structure of the app, and many other aspects of software being created. Concerning this concrete project there are some features that demand some specific approach to the thinking about its architecture. The main thing which should be mentioned is that this system is a combination of hardware and software components. These two components should work together to control the robot. When it comes to the discussion of software architecture in available sources (of software engineering) major part of authors assume that software is to be created for some enterprise or an office and almost never discuss peculiarities of systems which involve significant usage of hardware components. Obviously, such approach is not suitable for my project. The reason for such statement is that software robotic systems differs from other types of software (especially from standard desktop applications, mobile applications, and web-applications) because implement coordinated real-time interaction of non-heterogenous environments (electronic and mechanical in the case of my project).

According to analysed sources devoted specifically to software architecture for robotics (Ahmad & Ali Babar, 2016; Gomaa, 2000; Kortenkamp, 2015) the nature of robotic systems demands creation of specially designed software architecture. So called layered robot control architectures are widely used for design of software for robotic systems (Kortenkamp, 2015). For this project I learned two implementations of this approach which are two-layered (Coupled Layered Architecture for Robot Autonomy (CLARAty)) and three-layered architecture. The authors of mentioned sources show examples of successful practical usage of different types of software architectures and their variations for some real projects.

Robotic software three-layered architecture is presented by three levels: planner, executive, functional. This architecture is independent from concrete programming languages, frameworks and other specific underlaying software and can be implemented by needed and/or available tools. On the base of mentioned sources and analysis of demanded functionality of this project I've decided to follow conception of three-tiered architecture for robotic software. Reasons for this choice are given below.

Table 13: Correspondence between elements of three-layered architecture and structure of this project

Layer/ applicability for the project	Short formal description (Kortenkamp, 2015)	Possible representation in my project
Planner layer (highest level) / +	This layer serves for implementation of long-term goals of the system. This is set of high-level components which can create tasks (or compound commands) for execution and send them to components of executive layer.	This layer in my app can be implemented by set classes which form commands in real-time manner (but without planning, because such functionality is not needed). In my application the role of planner is played by modules which catch specially coordinated eyes movements of the user and his current EEG data. On the base of received information components of this layer can create commands of high level not concerning about details of their implementation. Strong benefit of this approach is that classes in this layer are not coupled with the equipment being used. Possible drawback that it is easy to create some implementation which is not connected to reality (I mean that components could demand from hardware something which is not possible). To mitigate possible negative effect, we need to know details of used hardware while programming components of this layer and follow its functional features.
Executive layer (middle level) / +	In general, this layer is described as a set of executable modules which can be used to choose	Each motion command of the robot comprises sequence of relatively simple instructions which describe signals needed to be sent to the device in order to get the required motions. In this

	current needed behaviour on the base of received tasks.	case I am prone to think that structure of my app complies with the rules of this layer. I need to define such classes which should form commands in required order and send them to the robot. This classes should operate over remote channels and create commands of used protocols according to syntax of used low level interfaces (such as GPIO and SSH). This implementation can allow me to program interaction between desktop app and control software on the side of the robot. This approach has some benefits for my project. This architecture allows to design relatively independent set of classes which could be used as a middleware in different projects because its classes don't depend from planner layer. In fact, planner layer doesn't depend on this layer too. So, we can say that this quite decoupled architecture.
Functional layer (basic level) / +	This layer Implements behavioural control and closely connected to sensors, actuators, motors and so on.	Concerning my project, this layer consists of software components which are provided by operating system of the robot's computer. They are capable to receive commands from remote sources and execute them reporting the result of execution afterwards. My task in this case is to configure system basic software for receiving data from my application.

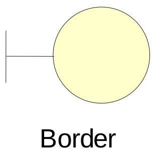
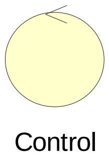
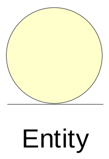
According to made analysis I can conclude that following this three-layer allows my project to comply with common recommended practices of software design such as separation of concerns (because class of each level should be responsible for concrete functionality), loose coupling (because classes of each layer don depend of each other).

5.1.7 The project structure: classes and components

Overall classes description will allow to design the app according to strict logical structure. During this stage of software development, I designed classes having SOLID principles of object-oriented programming in mind. Following class diagram (see Figure 12) show final design of classes and uses "Entity-Control-Boundary" stereotype of UML class diagram (Booch, 1991). I used this type of description of classes because it allows not only to show classes itself but also allows to demonstrate abstract role of the class in the app being designed. Boundary classes are those which interact with users of the system (or other external systems). Set of such elements of architecture can be shortly names as “front end”. Also, they create and send commands to classes of “control” type. *Control classes* are needed to execute commands from boundary classes and also, they are used to update state of entity classes. So, control classes can implement some sort of the flow of interaction between other elements of the app’s architecture. *Entity classes* are usually passive. They model persistent data used by application (any kind of data storage can be mapped to entity classes). Also, entity classes can be used itself as a data storage for the application.

Each of mentioned stereotype has its own graphical representation (instead of standard rectangle of standard class diagram). Used shapes are explained in the Table 14. They are used in class diagram of Figure 12 on page 48.

Table 14: Description of modules and classes

Border class	Control class	Entity class
		

Such quality characteristics as maintainability, portability, efficiency and others depend on well-designed modular structure of the software. Classes of the app are separated into different modules according to their functionality.

Table 15: Description of modules and classes

Module name and its description	Class name	Description
StreamVideo (dll) contains classes which implement receiving of the video stream from the robot’s camera and process it.	`GazeCommanderPictureBox`	This class receives and shows network video input, combines it with data from eye tracker and reports about target region of the user’s gaze.
	`MotionDirectionChangedArgs`	This class contains data of the event when direction of user’s gaze was significantly changed.
StreamGaze (dll) contains classes which implement receiving and processing of data from eye tracker.	`IGazeStream`	Interface for implementation of gaze input in the app. Can be used in order to implement ability of retrieving gaze input from different types of eyes trackers.
	`TobiiGazeStream`	This class represents the stream of data (coordinates of a point where gaze is located at each moment of time) produced by Tobii eye tracker. Also, the class reports of moment of fixations.
StreamMind (dll) contains classes which implement receiving and	`BrainActivityValues`	This class can be used as the source of some specific important characteristics

Border

Control

Entity

processing of data from EEG headset.		which influence the process of analysis of brain data.
	`BrainWaveEventArgs`	This class can be used as means of transferring of data about EEG headset device.
	`MindStream`	This is model of stream of data which is formed by EEG headset.
SettingsStorage (dll) contains classes which provide other classes of the app with persistent data storage functionality for static data used for their inner aims.	`SettingsData`	This is a set of characteristics of different objects of the app.
	`InternalStorageProvider`	This class is used as an interface between this app and operating system's services providing data storage controlled by OS.
GazeSurface (dll)contains classes which model an area capable to accept user's gaze and report about gaze events happening inside them.	`GazeSurface`	This class represents rectangular area for implementation of eyes movements detection.
Enumerations (dll) contains classes which have sets of constant named values.	`EnumStrings`	String titles of gaze rectangles.
	`MoveGoalRectangles`	Symbolic names for referencing of gaze rectangles.
ConnectorSsh (dll) contains classes which implement connection with the robot's controlling computer using secure shell protocol over network connection.	`SshConnector`	See description of the module ConnectorSsh.
ConnectorHttp (dll) contains classes which implement connection with remote computers over Http protocol.	`ConnectionStatusEventArgs`	This class transfer data about changes of network connection status.
	`HttpConnector`	This class implements network connection.
CommandSsh (dll) contains classes which implement creation of specific secure shell commands intended to be executed by robot's computer.	`SshGPIOCommand`	This class represents string form of some secure shell command.
CommandMotion (dll) contains classes which implement representation of robot's low-level commands in the form of methods of high-level programming language.	`IMotionCommand`	This interface models common idea of a robot motions and contains declarations of methods which can be used for concrete implementation of the in concrete classes.
	`GPIOSshMotion`	This class implements IMotionCommand interface and has methods for creation and sending of commands which should move the robot over SSH protocol and GPIO interface.
CommandGPIO (dll) contains classes which implement	`GPIOCommand`	This class represent low-level GPIO SSH command.

interaction with General Purpose Input Output system of Raspberry PI computer (which control DC motors of the robot).	`GPIOWheelPort`	This class models pin connectors of Raspberry PI breadboard.
ClassesExtentions (dll) contains classes which implement additional specific functionality for existing standard classes of .NET Framework.	`PointExtention`	This is an extension of Point class.
	`NewtonJsonReaderExtention`	This is an extension of JsonReader class.
	`StringToRobotCommandExtention`	This is an extension of String class.
GazeMindDriver (exe) represents main app of the system. Shows main window and creates event handling loop.	`MainWindow`	Main window of the app. Contains UI, shows video stream and caches gaze input and EEG input, dispatches commands to the robot.
	`GazeRectangleMotionAdapter`	This class concerts inner representation of gaze input to the invocations of the robot's commands.

The primary idea of each module is gathering together elements of similar functionality in order to achieve quite high level of module's components cohesion.

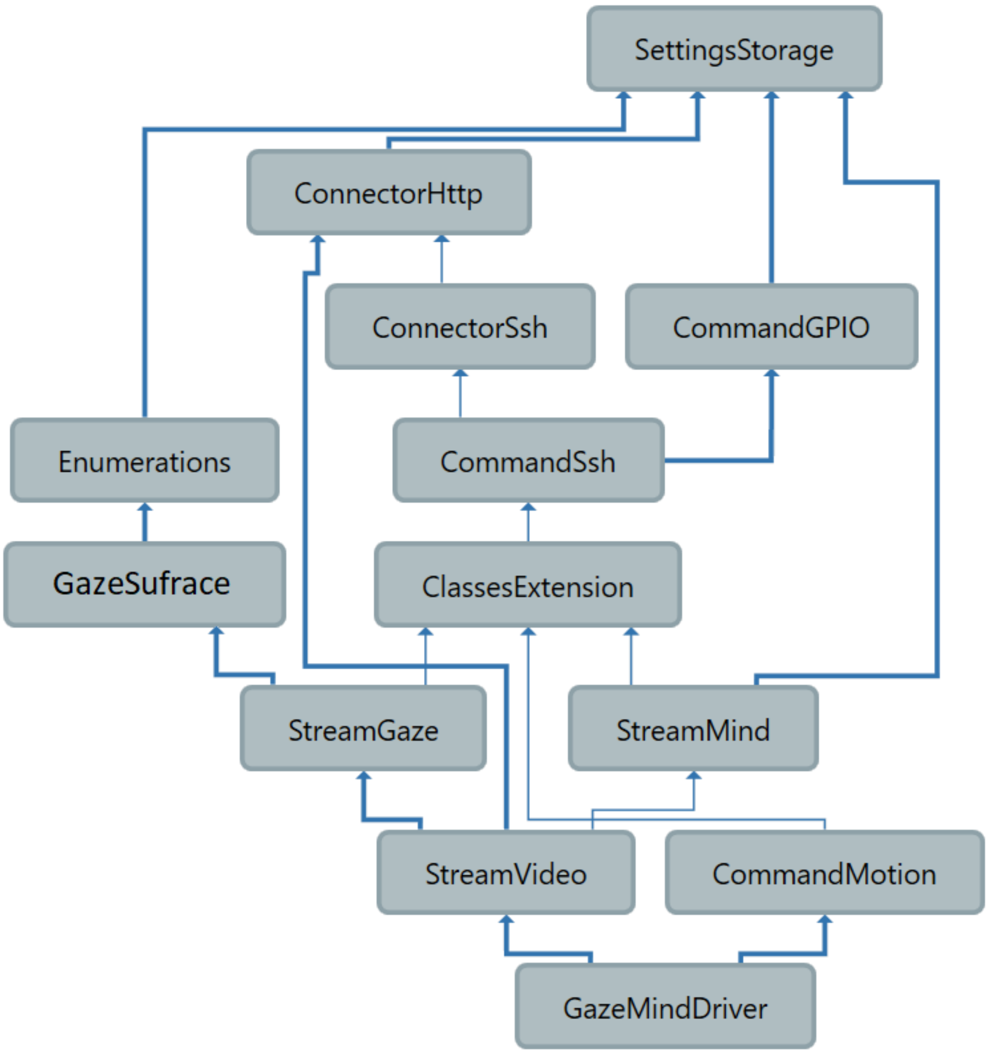


Figure 11: Modules dependencies diagram (arrows point to independent modules)

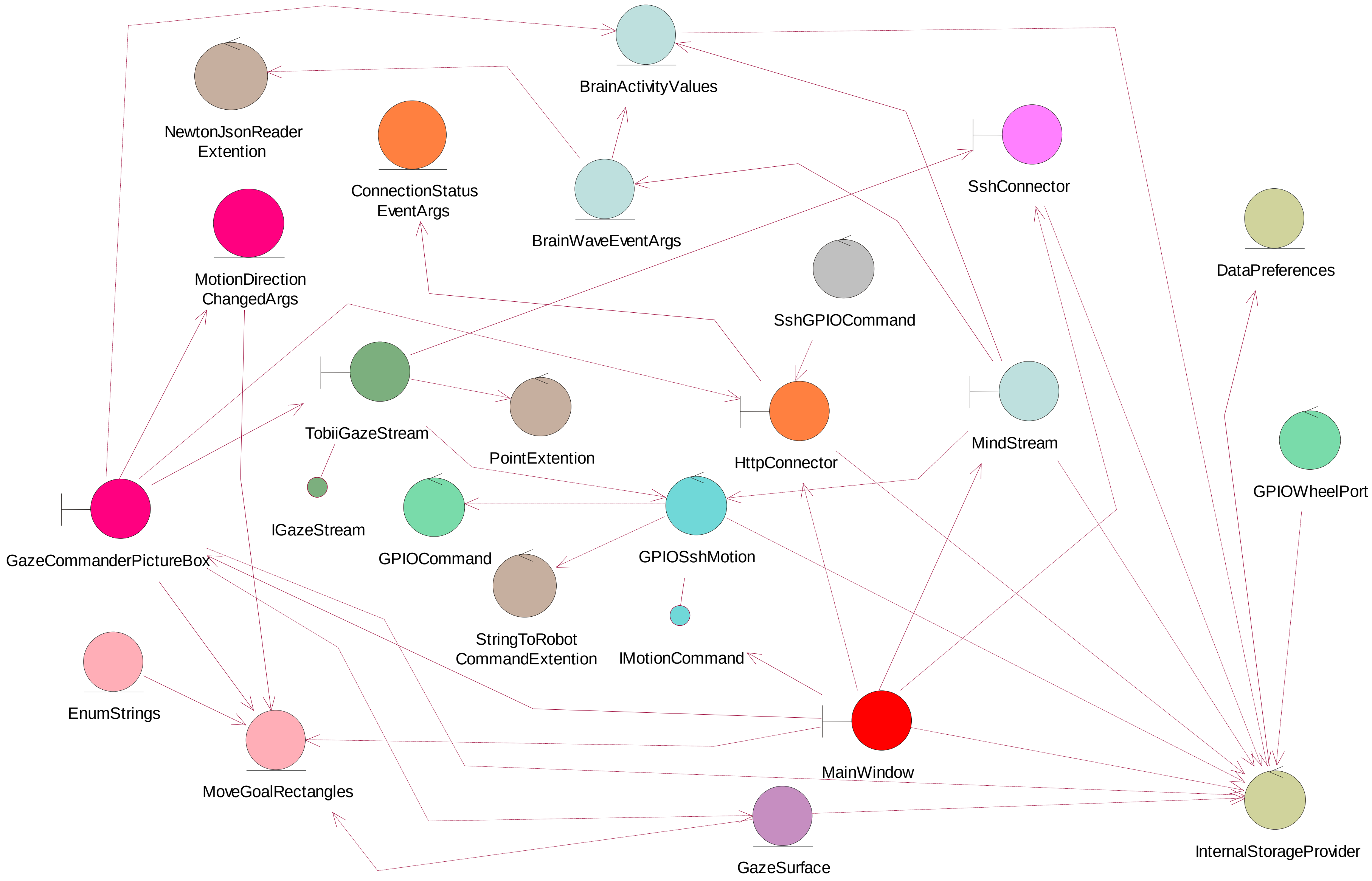


Figure 12: Final class diagram of the capstone project's app

Each class implements as narrow set of functionalities as possible in order to guaranty that changes in any of them entails minimum changes in other classes. Classes are prevailingly connected by dependency ("uses a") connections and association connection ("has a") connections. Also, some classes are derived from standard built-in classes of .NET Framework. Benefits of this structure and its possible pitfalls will be analysed in the chapter about the app's architecture.

The figure above shows the last version of the class diagram which evolved through rather long period of time. My first attitude to the problem was almost the same. But decomposition of classes was undertaken during the process of the app development. First rough version of my vision of the problem is presented by following figure.

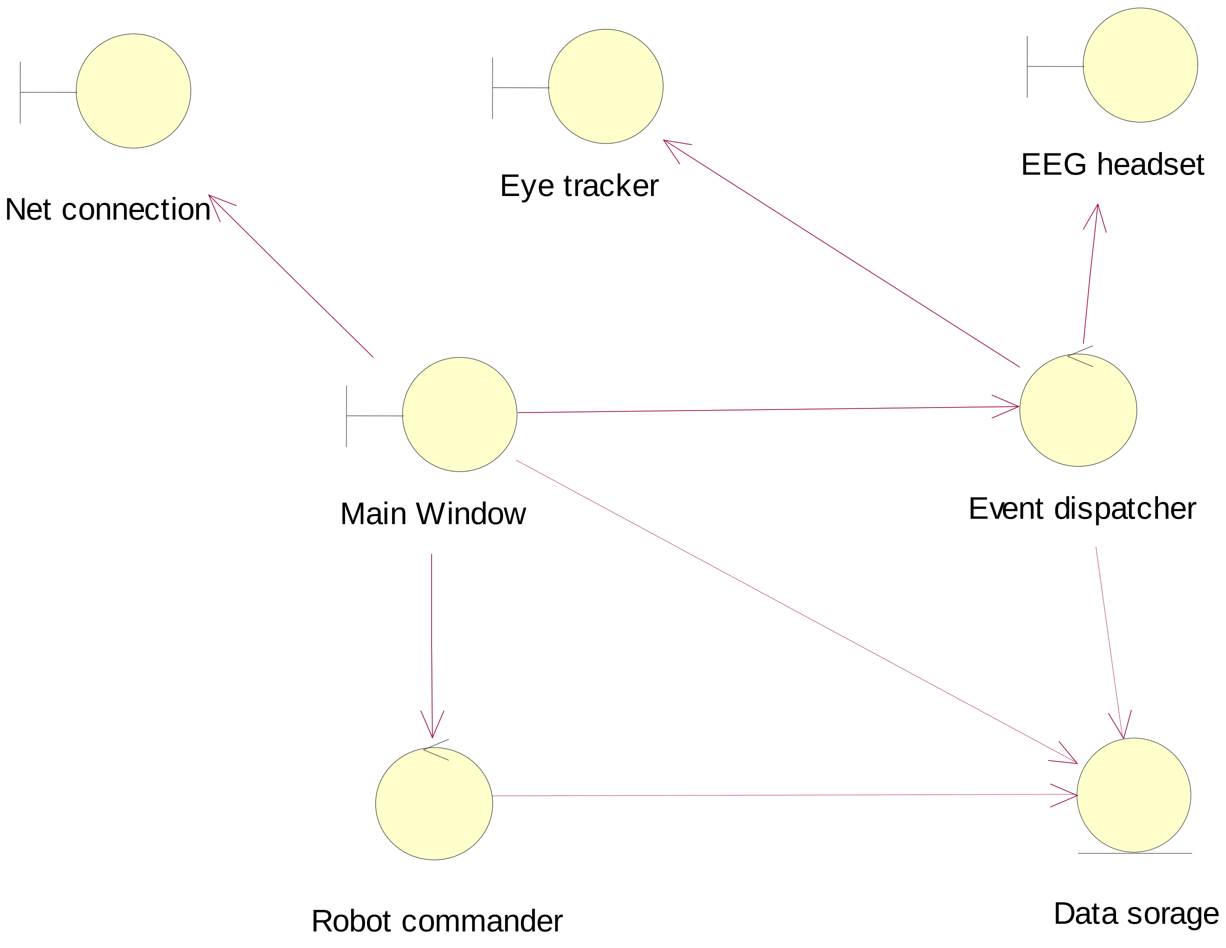


Figure 13: First class diagram of the capstone project's app

Significant changes were made because of deeper understanding of the ways which is better to use for interaction between devices (eyes trackers, EEG headset), PC and the robot. Used devices are provided with concomitant APIs and SDKs and their peculiarities influenced the final architecture as well. Description of used architectural patterns and design patterns used is given the next chapters of this report.

Also, it is needed to understand correspondence between used three-layered architecture for robotic software and created classes.

Table 16: Correspondence between layers of used architecture and modules of the app

Layer name	Level's components
Planner layer (highest level) Forming of high-level commands, sending of commands to the executive layer	SettingsStorage GazeSurface Enumerations CommandSsh CommandMotion ClassesExtentions
Executive layer (middle level) Forming of the equipment-oriented commands, interaction with the robot	StreamVideo StreamGaze StreamMind ConnectorHttp CommandGPIO
Functional layer (basic level) Physical execution of the commands.	This layer comprises of system services and components for interaction with the robot's equipment. Also, I created web-application for interaction with the robot's camera.

5.1.8 Behavior of the system

The next step of system development (according the rules of object-oriented approach) should contain modeling of system behavior. This can help to understand what should be done in the process of coding. Here I show some analytical UML diagrams which represent how process of interaction between used devices, the app and the robot is modeled in my system. I represent it with activity, state and sequence diagrams of UML (only the last versions of diagrams are presented). Each diagram depicts some specific aspect of the app functioning.

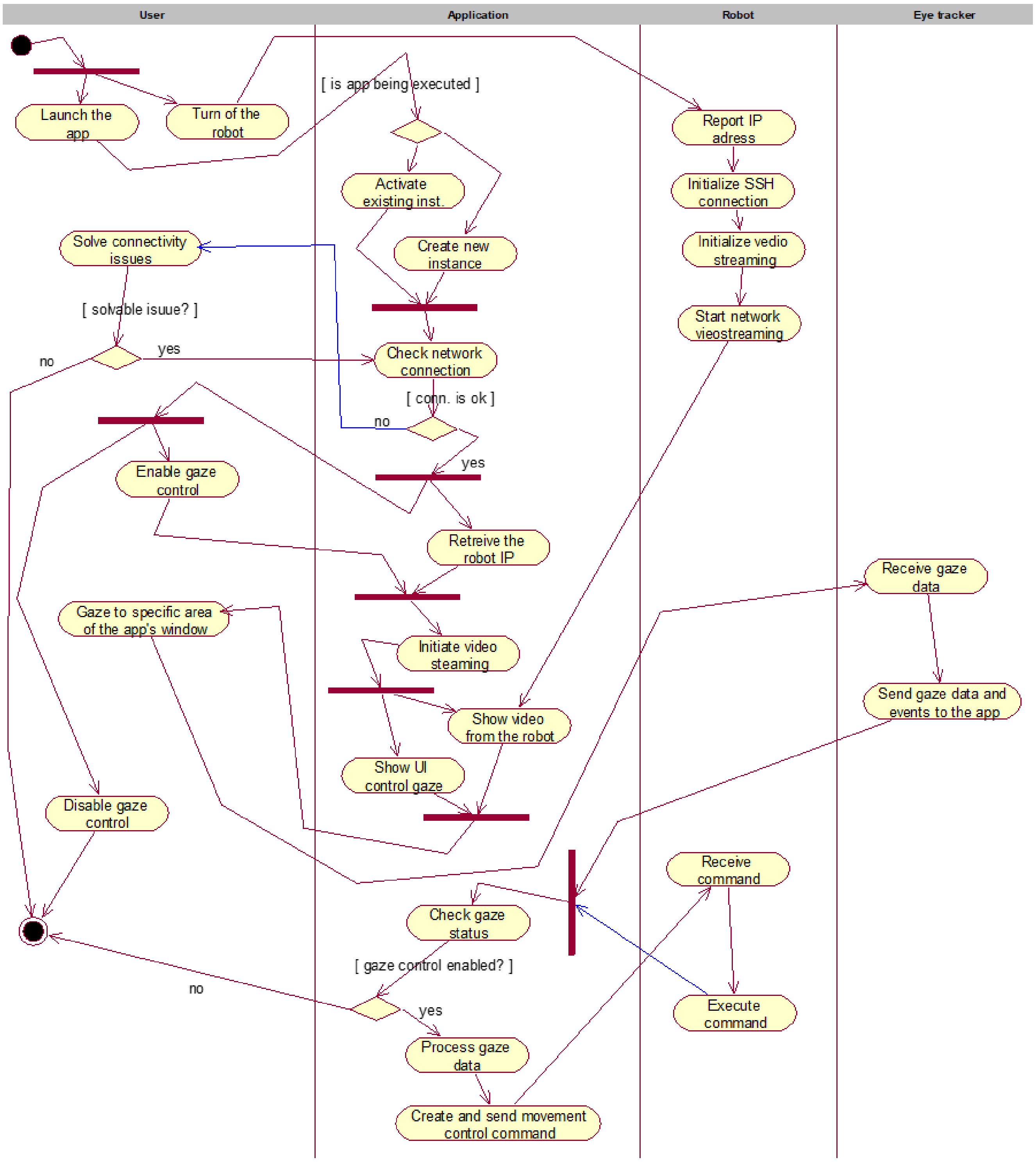


Figure 14: Activity diagram for use case "gaze control"

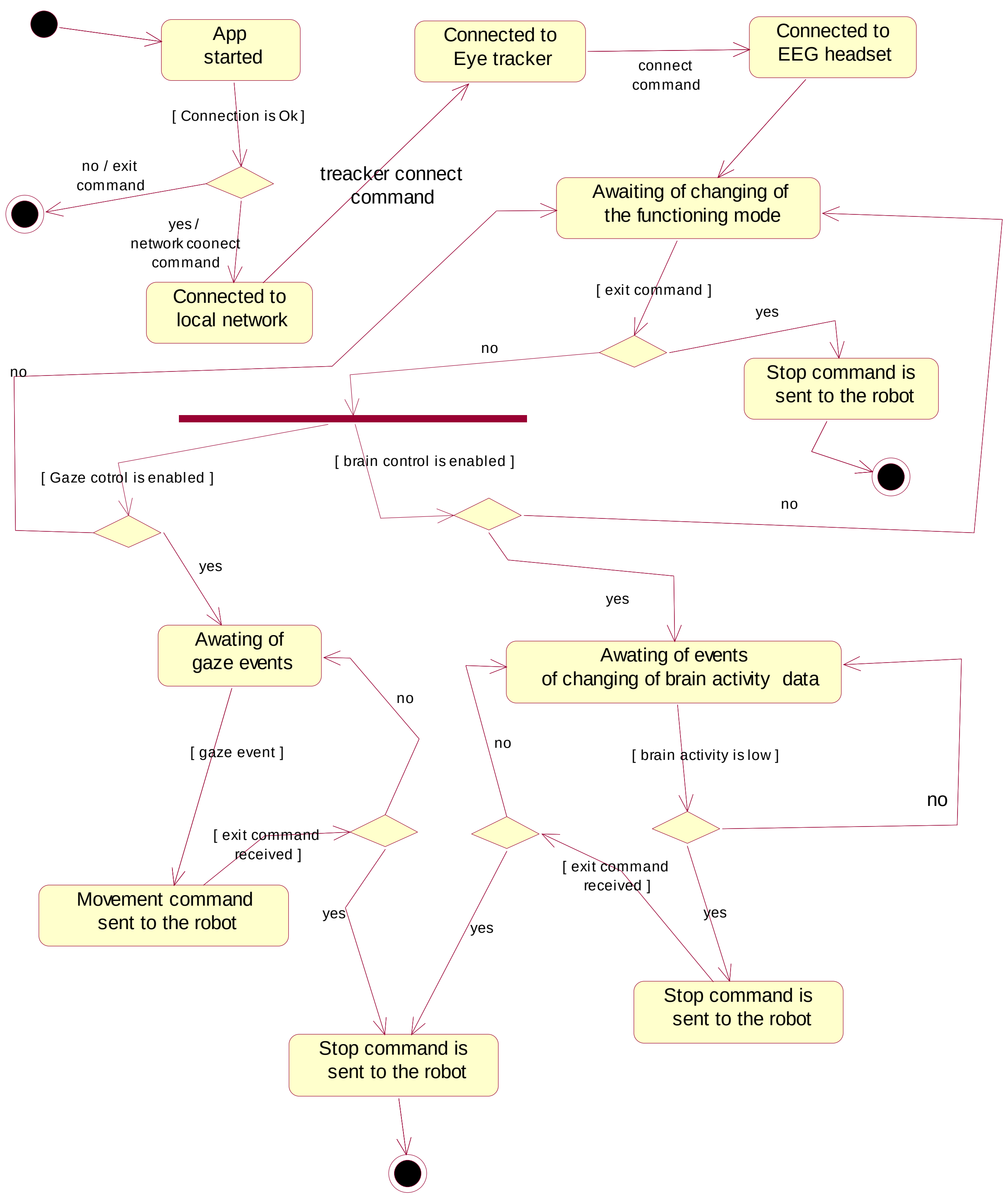

App started
Connected to Eye tracker
connect command
Connected to EEG headset
[Connection is Ok]
no / exit command
treacker connect command
yes / network coonect command
Awaiting of changing of the functioning mode
Connected to local network
[exit command]
yes
no
Stop command is sent to the robot
no
[brain control is enabled]
[Gaze cotrol is enabled]
no
yes
yes
Awating of gaze events
Awating of events of changing of brain activity data
no
no
[gaze event]
[brain activity is low]
no
[exit command received]
[exit command received]
yes
yes
yes
Movement command sent to the robot
Stop command is sent to the robot
Stop command is sent to the robot

Figure 15: The app's behaviour state diagram

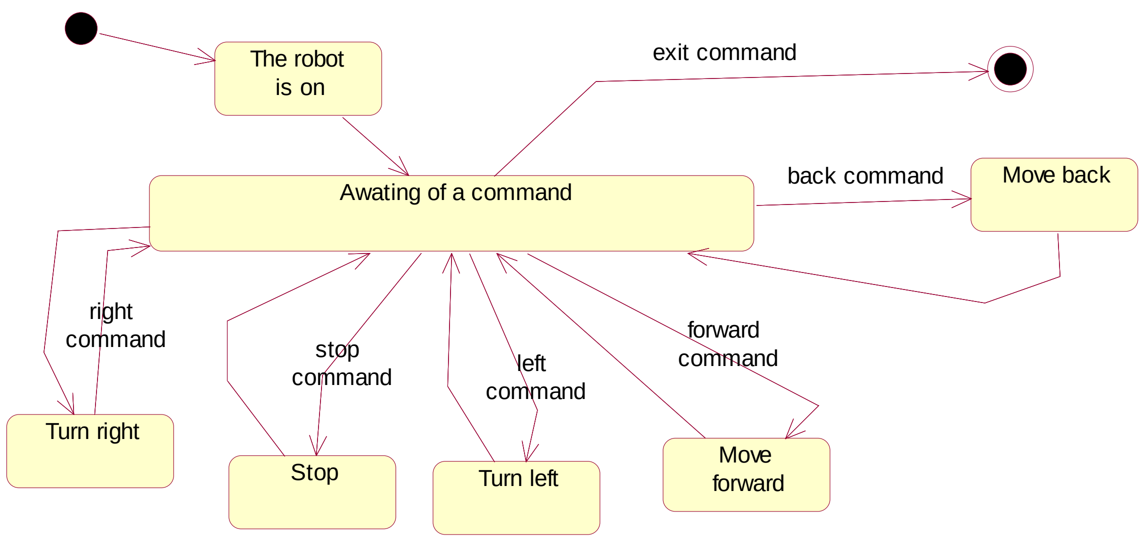


Figure 16: The robot's behaviour state diagram

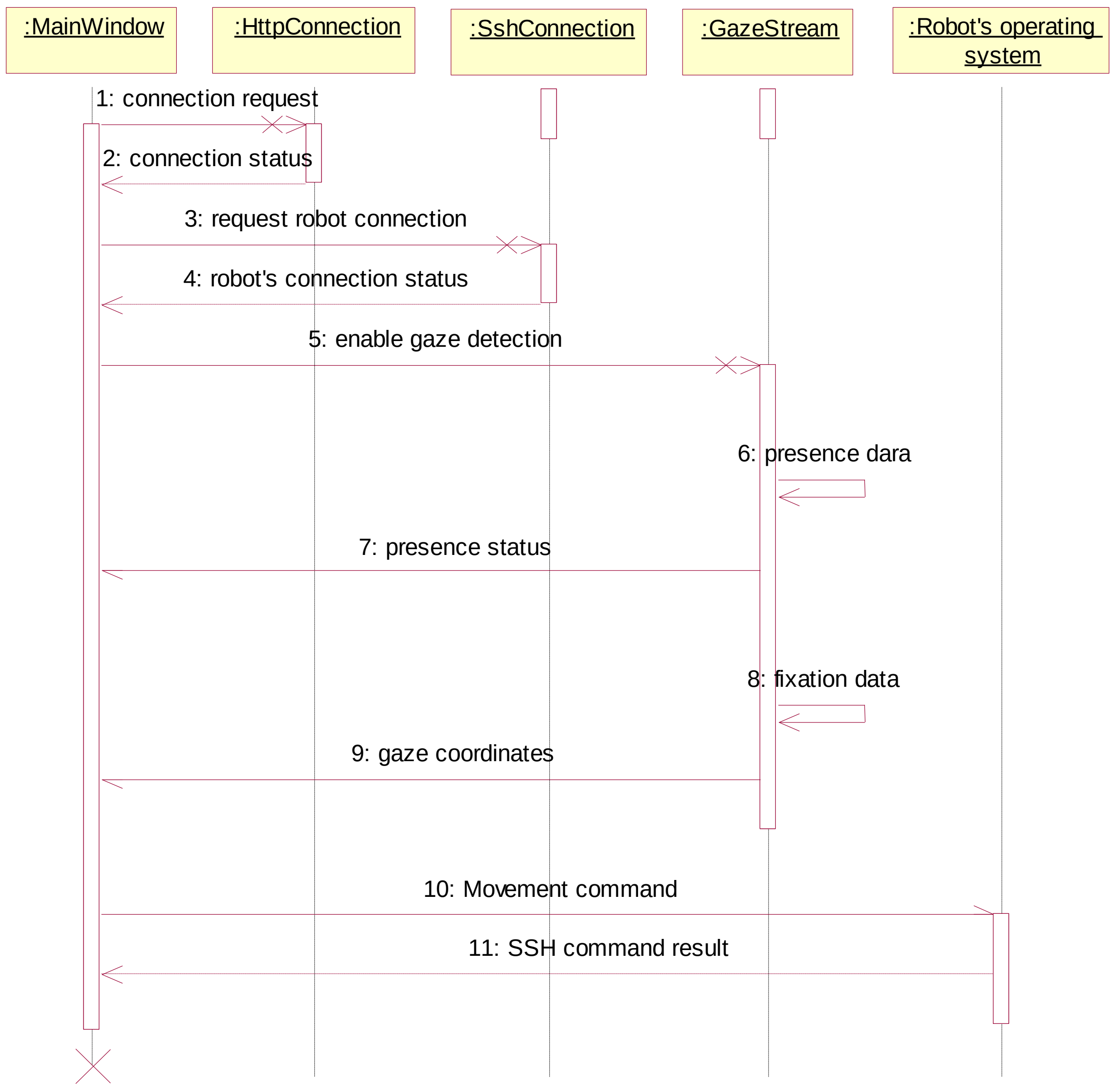


Figure 17: Sequence diagram of the app's robot control process

5.1.9 Reflection on activities of this section

Analysis of the task allowed me to understand both the overall structure of the planned software system and many details of it before I started to create source code using concrete programming language and specific development framework. Learning of principles of object-oriented modeling and usage of them during several iterations of development allowed relatively quick realization of needed changes. Usage of UML provided a way to express clearly the ideas of how the system should be organized and how it should function.

Difficulties which I encountered during the process of planning and analysis were mainly connected to some specific problems. Architecture of the system depends not only on requirements and needs of a user, but also on abilities and limitations of hardware part of the robot. This project was provided with hardware components which are capable to receive commands and execute them. But they lack ability of reporting the result of such execution. So, it is not clear whether some command was executed successfully or not. This circumstance didn't allow to plan and implement fully responsive system. This also leaded to some simplification of initially planned system architecture and behavior. But anyway, working in with components with limited functionality and trying to adapt the software to them was the valuable experience.

5.2 Requirements

Previous stage of the development gives possibility to elicit and formulate specific requirements to the software and hardware system being created (Table 17).

Table 17: Requirements for hardware and software parts of the system

HR – Hardware requirements (components of the system)	
HR-1	System should consist of the following hardware parts: 1. PC, 2. Eyes movements tracker "Tobii Eye Tracker 4C", 3. EEG headset "NeuroSky MindWave Mobile 2 Headset", 4. Movable robot.
HR-2	EEG headset should transfer data of user's brain activity through Bluetooth channel.
HR-3	Eyes movement detector (eye tracker) should be connected with PC by USB port.
HR-4	The PC being used should have permanent internet connection.
HR-5	The PC being used should have permanent local network connection with speed up to 100Mbs.
HR-6	The system should provide a robot which should contain following parts: 1. passive equipment: a. metal platform with holes for mounting of other parts of system, b. 3 wheels, c. jumper wires for connecting of electronical parts of the system, d. rechargeable battery. 2. active programmatically controlled parts: a. Raspberry PI computer, b. electric breadboard, c. L293D motor driving integrated circuit connected to two direct circuit motors and the breadboard,

	d. video camera.
HR-7	The robot should have permanent internet connection.
HR-9	The robot should have permanent local network connection.
HR-10	PC should be equipped with 1 gigabyte (GB) RAM (32-bit) or 2 GB RAM (64-bit).
HR-11	PC should be equipped with 1 gigahertz (GHz) or faster 32-bit (x86) or 64-bit (x64).
HR-12	PC should be equipped with 16 GB hard disk space (32-bit) or 20 GB (64-bit).
HR-13	Resolution of graphical system should be not less than 1920×1080.
HR-13	Recommended version of Raspberry PI is 3 or higher.
SR – Software requirements	
GSR – general requirements to basic software being used	
GDR-1	PC being used should be run under control Windows operating system (version 7 or higher).
GDR-2	PC should have proper Bluetooth driver installed.
GDR-3	PC should have proper eyes movement tracker's driver installed.
GDR-4	PC should have .NET Framework runtime installed (version 4 or higher).
GDR-5	PC should have NeuroSky ThinkGear© server software installed.
GDR-6	Raspberry PI should be run under control of Linux operating system (kernel version 4 or higher).
GDR-7	Raspberry PI onboard operating system should enable remote SSH connection.
GDR-8	Raspberry PI onboard operating system should enable remote GPIO control.
GDR-9	Raspberry PI should have Apache server installed and being executed.
GDR-10	Raspberry PI should have PHP interpreter installed.
GDR-11	Video camera of the robot should provide picture of sizes 1024 by 768 pixels.
GDR-12	EEG headset should transfer data through Bluetooth channel in raw binary format (binary packages) according to technical specification of this device.
GDR-13	Eyes movements tracker should transfer data packets through serial port connection.
SFR – Robot-side software requirement	
SFR-1	There should be a script which is capable to report local network IP address of the robot's computer.
SFR-2	The script should be launched at the moment of launching of Linux operating system.
SFR-3	The script should write IP address to cloud Firebase database in the standard string format ***.***.***.***
SFR-4	The robot should be provided with server PHP script which is capable to get picture from the robot's camera.
SFR-5	Server script should be accessible from local network by local IP address.
SFR-6	Server script should be accessible from local network by HTTP protocol.
SFR-7	Server script should provide (on request) the latest picture acquired from the robot's camera.
SFR-8	The response time to the image request should not exceed 20 ms.
SR – Desktop software requirements	
SRUI – Desktop user interface requirements	
SRUI-1	Desktop application should have standard graphical user interface according to Microsoft guidelines (Microsoft Corporation, 2018).
SRUI-2	The main window should have standard header with disabled maximization button.
SRUI-3	The main window should have border without ability of resizing.
SRUI-4	The main window should have fixed size during the period of its continuous working.
SRUI-5	The main window should form graphical area for video retrieved from the robot's video camera (this area is further referenced as video frame (VF)).
SRUI-6	VF should provide 6 specifically designated controlling rectangular areas which are capable to receive gaze events. Further such areas are referenced as GCR (gaze control rectangle). 1. Trigger GCR. 2. Forward GCR. 3. Backward GCR. 4. Turn Left GCR. 5. Turn Right GCR. 6. Stop GCR.

SRUI-7	The main window should provide standard toolbar for accessing of the app's functionalities.
	SRF – Desktop app functional requirements
SRF-1	Only one instance of running application is allowed for current Windows account.
SRF-2	The app should check the state of network connection during the time of its functioning. a. if connection is ok, then commands to the robot can be sent, b. if connection is broken, then control commands are blocked and not sent.
SRF-3	The application should store characteristics of its hardware and software components in external isolated data storage provided by Windows operating system in the form of XML file.
SRF-4	Each Windows user's account should have its own copy of the app's data storage file.
SRF-5	User's gaze at trigger GCR should turn alter the mode of control of the robot movements. a. If gaze control is on, then trigger GSR should turn it off. b. If gaze control is off, then trigger GSR should turn it on.
SRF-6	User's gaze at "Forward GCR" should force the robot to start moving to the direction where its camera is directed (forward moving).
SRF-7	User's gaze at "Backward GCR" should force the robot to start moving to the direction which is opposite to where its camera is directed (backward moving).
SRF-8	User's gaze at "Turn Left GCR" should force the robot to start moving left relative to the direction in which its camera is directed.
SRF-9	User's gaze at "Turn Right GCR" should force the robot to start moving right relative to the direction in which its camera is directed.
SRF-10	User's gaze at "Stop GCR" should force the robot to stop (if it's moving).
SRF-11	The application should provide the user with ability to use EEG headset for controlling of movement of the robot: a. if level of overall brain activity reported by EEG headset is lower than predefined value, then movement of the robot should be stopped, and control of the robot by the user's gaze should be prevented; b. if level of overall brain activity reported by EEG headset is equal or higher than predefined value, then movement of the robot should be allowed.
	PR – System performance requirements
PR-1	The application should provide the user with the latest picture from the robot's camera with possible latency not more than 1 second (under the condition that appropriate quality LAN connection is provided (HR-4)).
PR-2	System should guaranty immediate reaction of UI and the robot on gaze events (within defined temporal scope): a. if fixation inside of some GCR is reported, then command should be sent to the robot immediately (time of sending is no longer than 1 sec.).* b. the robot should start executing of received command not later than 1 second from the moment when this command was sent*.

*Estimation of a length of immediate reaction is based on researches (Nielsen, 1993) of peculiarities of human subjective perception of time frames of system's UI responsiveness.

5.3 Reflection

Activities of this stage of the project were repeated several times during the process of the system development because of usage of incremental iterative SDLC. Improvements in models reflected my new understanding of some aspects of the hardware and software functions. Analysis of the system's aims and its possible logical structure were quite both interesting and some sort of challenging.

Main problems came from peculiarities of needed functionality in hardware part. I mean that resending of user's gaze data to the robot in the form of executable commands demands implementation of well-synchronized system of software and hardware working together and this is the thing which is very important to catch on the stage of analysis. It was not possible to clearly comprehend from the scratch what software components should be created and how they should interact. Only practical experiments and explorations allowed me to deepen my understanding of the abilities of hardware part of the system. When I understood capabilities of hardware part of the system, I managed to create complete and expedient architecture of the system.

Although RUP is recommended for relatively large business IT projects, its usage in this project allowed to utilize strict logical procedures of this approach and base the project on firmer logical basement. Also, object-oriented approach and UML gave benefits of clear description of the system (which eased process of coding, because created model was explicitly mapped to the classes of the app).

6 SYSTEM DESIGN AND IMPLEMENTATION

6.1 Discussion of software and hardware components of the system

According to previously described architecture (figure 1) and requirements (table 17) now I should analyze and choose suitable devices and other relevant hardware components to build this architecture. Also, I needed to choose appropriate data exchange protocols and software development tools.

6.1.1 Discussion on hardware components

Implementation of needed system's functionality demands usage of some specific hardware components. Nevertheless, this capstone project is not devoted to hardware architectural analysis and implementation. Because of this circumstance I am not going to deeply describe process analysis and comparison of used hardware components. They were chosen in the process of communication and cooperation with the project's coordinator who has strong expertise in this area of engineering.

Table 18: Description of main hardware components

Component name	Component type	Role in the system
Tobii Eye Tracker 4C.	Eyes movement detector.	Eyes movements tracker is used to provide system with information about where a user's gaze is directed. Literally this device plays the role of steering wheel which can be controlled by eyes of the user.
NeuroSky MindWave Mobile 2 Headset.	EEG brain activity detector.	This device provide system with information about how active the user's cognitive processes and activity. This information is needed to take decision whether movement of the robot can be continued at each moment of time while the system is in use. If level of brain activity drops under some specific level, then this fact can be interpreted as situation of significant distraction of the user from the process of driving. In this case the robot should be stopped until previously sent movement command is confirmed by repetitive movements of eyes.
L293D.	Motor driving integrated circuit.	This component is used to programmatically control DC motors of the robot (IncludeHelp, 2019). This device provides the system with ability to rotate wheels attached to the used DC motors. This chip can control up to two motors and regulates velocity of such rotation.

DG02S Mini DC GearBox.	Direct current motor.	This type of devices are rotary electrical machines which is used to converts DC energy into mechanical energy. Used DC motors realize movements of the robot.
Raspberry PI 3	Personal compact computer.	This part is used as a control center of the overall robot behavior. This computer receives commands from the desktop application, interprets them and sends them for execution to the motors using GPIO.
Bluetooth USB dongle.	Receiver of wireless Bluetooth signals.	Bluetooth receiver is used to get signal and data from used EEG Headset. This device is used on the PC by drivers of EEG headset to provide connectivity of this device with its clients.

Choice of mentioned components were made on the base of their expedience. All parts of hardware were already in possession of the faculty by the beginning of the project (except Tobii Eye Tracker 4C) and had been used earlier in projects of other students.

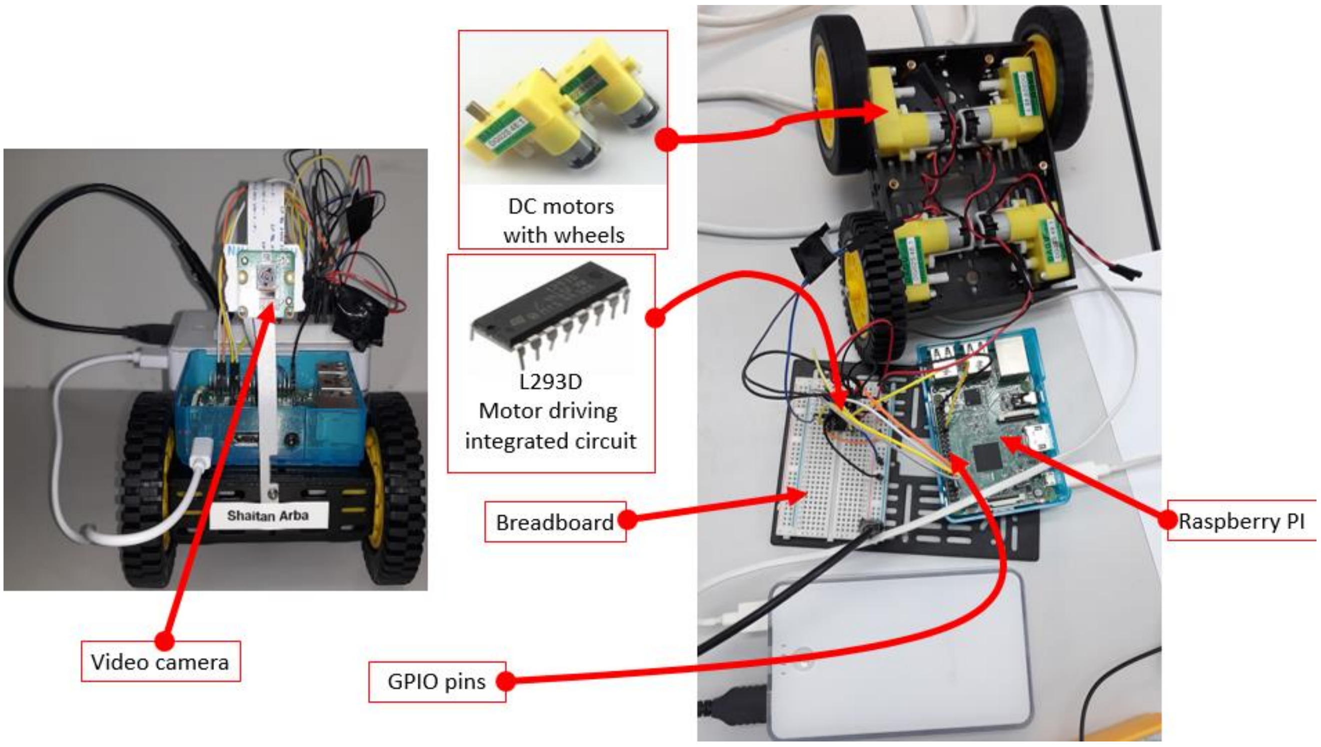


Figure 18: Description of the robot architecture

A circuit diagram on figure 19 (Silikoid, 2019) shows how outer components (the motors and the controller) that should be connected to the pins of Raspberry PI computer.

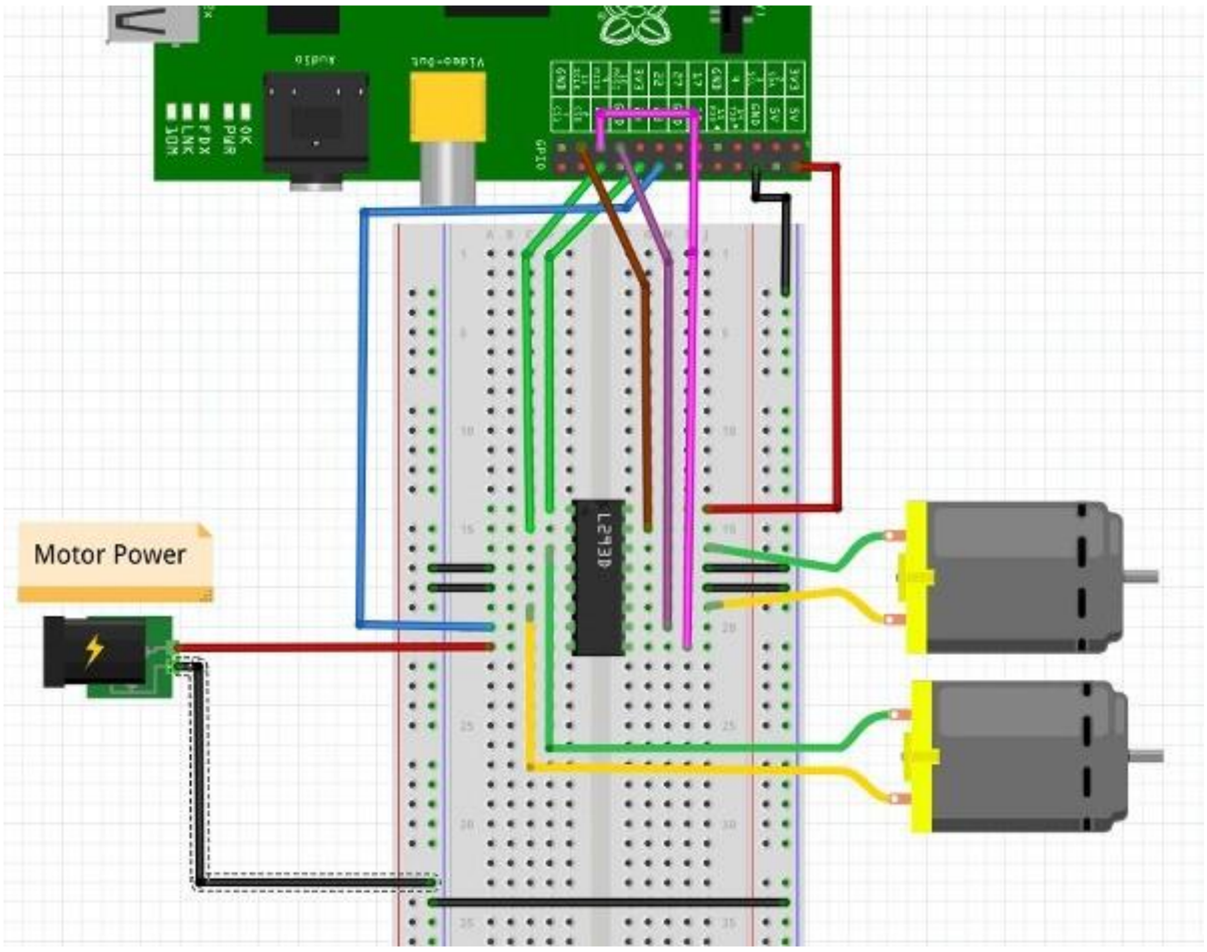


Figure 19: Circuit diagram of hardware part of the project (Silikoid, 2019)

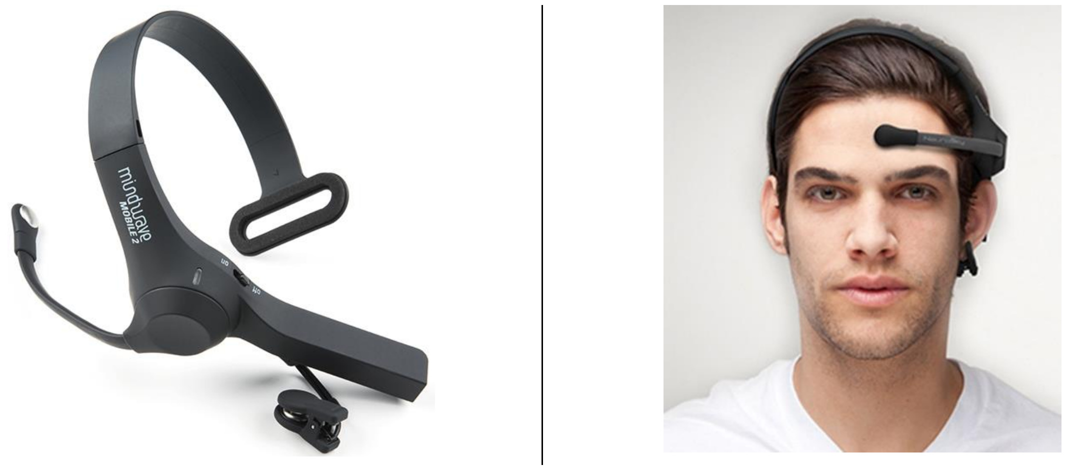


Figure 20: NeuroSky MindWave Mobile 2 Headset

Device of BCI for this project was chosen in my previous project for "730 Robotics and IoT" studying course on the base of the analysis and comparison of several available options. They are Emotive EPOC devices (Emotiv Inc., 2018), InteraXON Muse devices (InteraXon Inc., 2018), Neurosky MindWave devices

(NeuroSky, Inc., 2018). All of these devices use electroencephalography as a source of data and are based on usage of outer electrodes attached to subject’s scalp. Resulting preciseness and reliability of brain’s activity data depends from used number of sensors, their sensitivity and characteristics of algorithms being used inside devices to calculate and interpret final results of measurements.

If to tell about eye movement detection hardware it is can be noticed that there is only one major supplier of such kind of equipment on modern market. This is Tobii company which designs and manufactures quite wide range of devices for implementing of oculomotor interface.

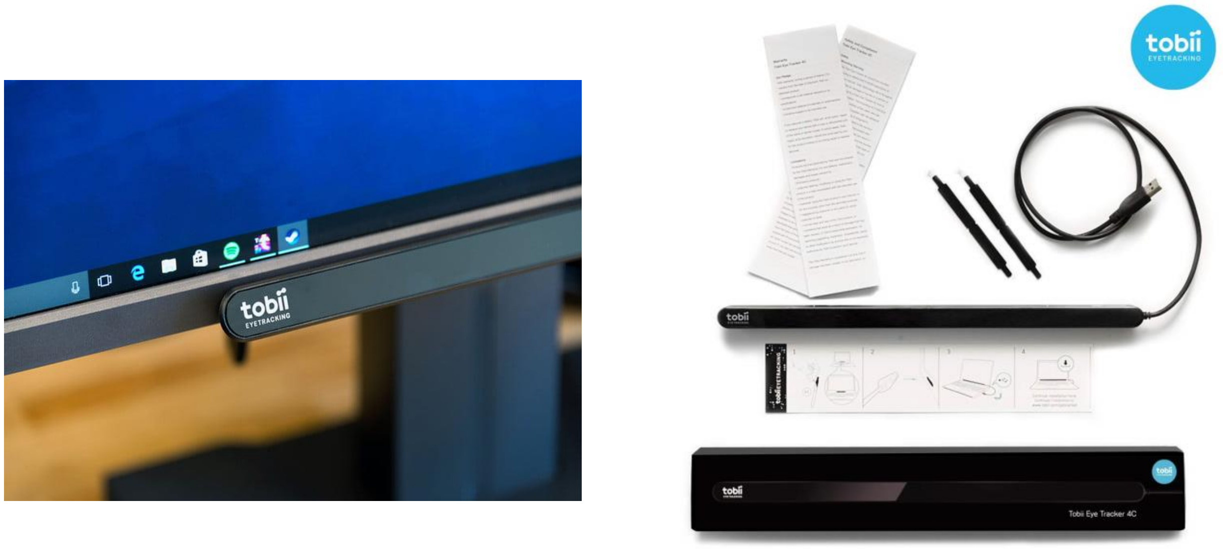


Figure 21: Tobii Eye Tracker 4C

They provide both professional systems and relatively basic devices which differ from each other by complexity and provided functionality. Anyway, eyes movement detector of this company is rather sophisticated apparatus (which contains sensors of some different types and allows to detect eyes of a user, his presence or absence, tilts and rotations of his head). Used “Tobii Eye Tracker 4C” is a device of consumer level (not professional) and has some limitations. First of all, this device can used only on computers with Windows 8 or 10 operating system (other operating systems, such Linux or MacOS are not supported). Also, this device is suitable only for desktop systems (not mobile) and uses wired connection (USB).

Possible alternative technologies in this area are based on usage of video registration of eyes movements (by web-cameras or video cameras). Such approach was not used in the project because this technology has some *disadvantages*. It demands very stable posture of the user in front of the computer (some systems fix user’s head with special holder and doesn’t allow any movements besides movements of eyes). If a user doesn’t follow these limitations and moves his head freely, then precision of detected eyes movements cannot be guaranteed. Also, there are some devices which can be used for the purposes of the project. One of them is Microsoft HoloLens headset (and other analogical VR-helmets and goggles) which is rather advanced mobile device, but its price starts from $3500 (this is unacceptable for this learning project).

6.1.2 Discussion on robot's software components

Software components of the robot were implemented considering the objectives of the project and its requirements.

Table 19: Description of robot's software components

Robot's software component	Function of the component	Used technologies
Start script RaspIPWriter.py	Retrieve IP address of the Raspberry PI computer and send it to Firebase DB for exchange with desktop application.	Python programming language and its standard library. Linux BASH scripting.
PHP page http://IP_ADRESS/ /html/cam_pic.php	Retrieve video stream from the video camera of the Raspberry PI and redirect it to the published folder of Apache server. This part of the system works as continuous HTTP service.	PHP, HTML.

Table 20: Description of robot's software technologies

Used technology	Possible alternatives	Function of the component in the system	Reason for the choice of used technology
Apache Web-server (Peicevic, 2016)	Nginx	This software is used to host server-side web-application for implementing the interaction between desktop app and the robot (major task is handling http requests for video stream).	Apache server is provided with more third-party components for implementation of many kinds of functionality. Particularly, there are some modules for interaction with Raspberry's video camera for Apache. Also, Raspberry PI is a computer with limited computational capabilities. In this case Apache also is more suitable.
PHP (ver.7) (Ajzele, 2017)	Java Server Pages (JSP)	Retrieve video stream from the video camera of the Raspberry PI and redirect it to the published folder of Apache server. This part of the system works as	Both, PHP and Java can be run under Apache server and send http responses to network clients. But PHP consumes less resources of computer (memory and processor time) though has lower level of performance (Trent, Tatsubori, & Suzuruma, 2019). In the case of Raspberry PI this circumstance plays leading role because affects energy consumption. Robot is standalone device which gets energy from mobile power bank. Time

		continuous HTTP service.	of its functioning significantly depends from how much resources software components consume.
RPi_Cam_Web_Interface library (Elinux.org, 2019)	VLC media server	This software is a set of APIs which provides direct connection to video camera. It provides with ability to configure characteristics of video being retrieved (size, quality, rotation, frequency of updating and so on).	The main advantage of the API used is that it has much less latency of the resulting video stream. Other media servers create video stream which has significant latency (1 second and even more). This is absolutely unacceptable because the system needs real-time online video stream. Slight disadvantage of this library is that it emulates video by saving consecutive still images to the same file in the same folder (this is emulation, but not real video stream). But it doesn't matter for overall outcome because efficiency is quite high and latency is relatively short (not longer than 1 second, which is needed according requirements).

6.1.3 Discussion on desktop software components

Major time of the system development was devoted to the desktop application which is intended to be used as controller of robot movements. The app being created should be some kind of console of the robot. It must accept user's gazes and his brain activity data and transform all this data to movements commands for robot. This task was implemented by usage of the following software components and technologies: .NET Framework and Visual C# programming language (Troelsen, 2017).

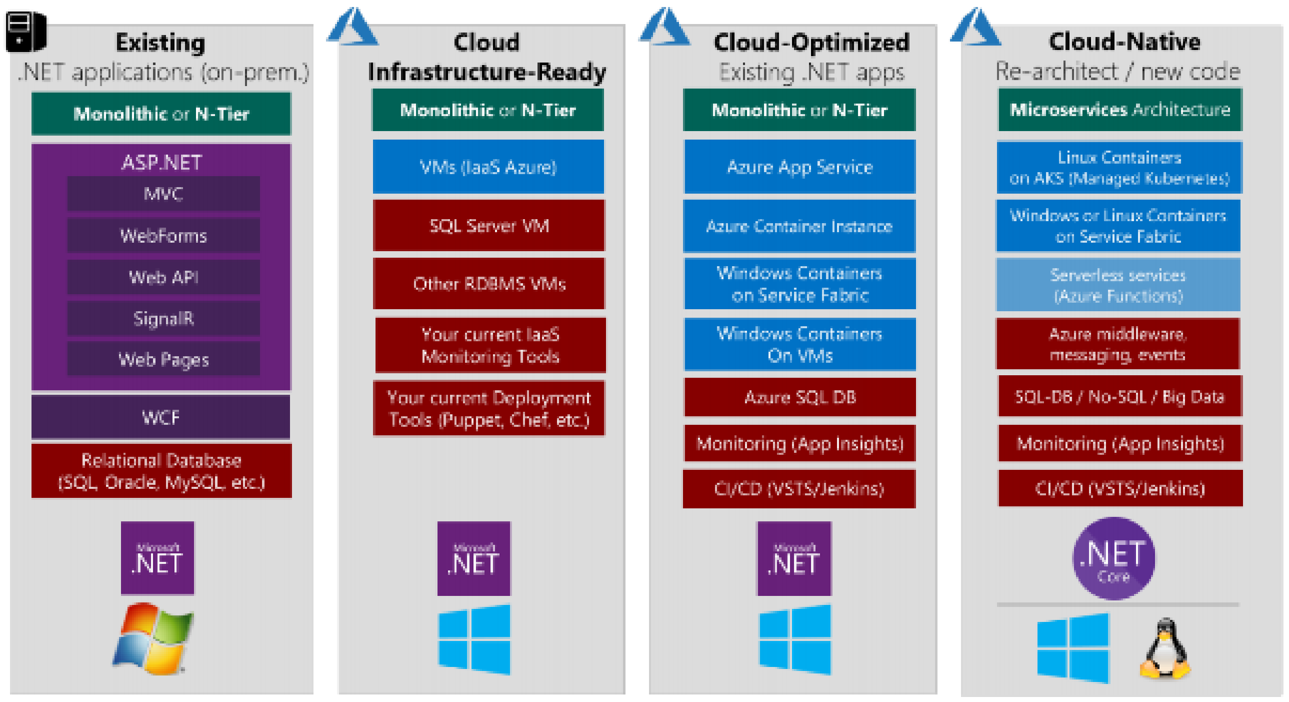

Existing
.NET applications (on-prem.)
Monolithic or N-Tier
ASP.NET
MVC
WebForms
Web API
SignalR
Web Pages
WCF
Relational Database (SQL, Oracle, MySQL, etc.)
.NET
Cloud Infrastructure-Ready
Monolithic or N-Tier
VMs (IaaS Azure)
SQL Server VM
Other RDBMS VMs
Your current IaaS Monitoring Tools
Your current Deployment Tools (Puppet, Chef, etc.)
.NET
Cloud-Optimized
Existing .NET apps
Monolithic or N-Tier
Azure App Service
Azure Container Instance
Windows Containers on Service Fabric
Windows Containers On VMs
Azure SQL DB
Monitoring (App Insights)
CI/CD (VSTS/Jenkins)
.NET
Cloud-Native
Re-architect / new code
Microservices Architecture
Linux Containers on AKS (Managed Kubernetes)
Windows or Linux Containers on Service Fabric
Serverless services (Azure Functions)
Azure middleware, messaging, events
SQL-DB / No-SQL / Big Data
Monitoring (App Insights)
CI/CD (VSTS/Jenkins)
.NET Core

Figure 22: Microsoft .NET Framework technology architecture (Microsoft Corporation, 2017)

Creation of this desktop application is based on the principles of object-oriented programming and uses pure object-oriented technology which is called Microsoft .NET Framework. Today object-oriented stack of technologies (such as object-oriented analysis, object-oriented modeling, object-oriented design, object-oriented programming, object-oriented databases) are actively being evolved and are widely used for creation of different types of software. Concerning .NET Framework technology which was utilized in this project, it is needed to say that this is a large stack of underlying technologies (Villela, 2019). These technologies provide very wide range of functionality for creating almost any type of software. This can be cross platform desktop applications, web applications, mobile applications, server applications and so on. Diagram of figure 22 shows structure of .NET technology.

Along with .NET Framework other object-oriented technologies can be taken in consideration. Here I would like to mention Qt framework (Sheriff & Lazar, 2018). This technology emerged much earlier than .NET and is developing by different companies since 1995 (Trolltech, Nokia, Qt Project, Digia, The Qt company). This technology is based on C++ programming language (Stroustrup, 2013). I consider this technology firstly because manufacturers of used eyes tracker (Tobii) provide several implementations of native API for their devices. Namely they have versions for .NET Framework and for C++ programming language and its runtime.

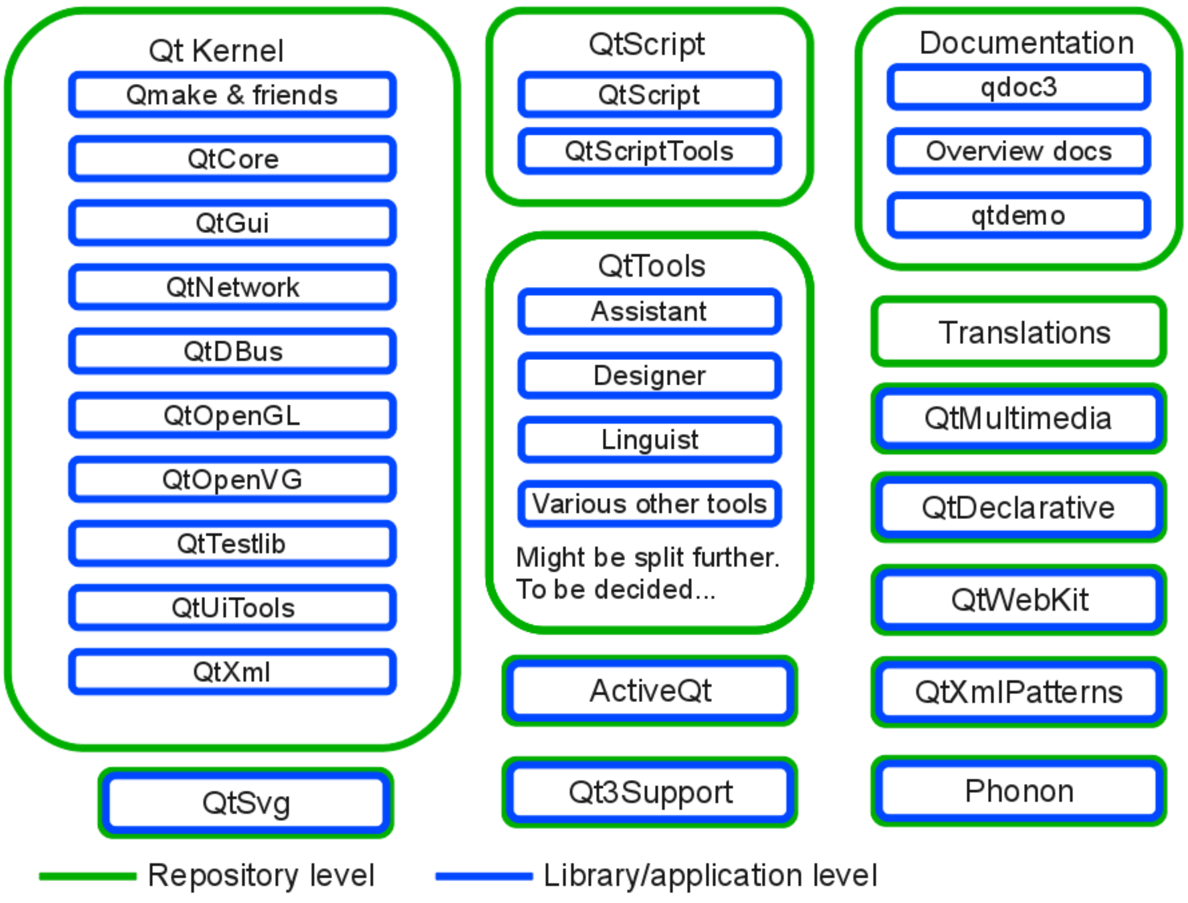


Figure 23: Concise graphical representation of Qt structure

Qt is a technology which provide programmers with huge cross-platform class library. Qt has realizations for all modern operating systems (Windows (desktop and mobile), Linux, MacOS, Android, iOS, and embedded operating systems). Software which developed against this framework can be written once and compiled to native executable code of any supported operating system. Shortly architecture of the technology is shown on figure 23 (Sheriff & Lazar, 2018).

If to compare these technologies, we can see more similarities than differences (table 21). Result of the comparison on the base of supported and implemented possibilities is shown in the table 21 (The Qt Company, 2019; Microsoft Corporation, 2018).

Table 21: comparison of Qt and .NET Frameworks technologies

	Qt	.NET Framework
Full support of classical OO-programming concepts	Implemented in C++ language.	Implemented in C# and Visual Basic languages
Desktop applications	Implemented in Qt Widgets and QM.	Implemented in Windows Forms, Windows Presentation Foundation
Data access and data modeling	Implemented in QDataBase, QSQL and so on.	Implemented in ADO.NET, Entity Framework, LINQ.
Web applications and Web-hosting	Not implemented.	Implemented in Internet Information Services, Azure, OneDrive.
Web scripting	Implemented in Qt Web Engine, Qt Script.	Implemented in ISAPI, HTTP Handler, HttpModule.
Remote invocation	Implemented in Qt Remote Objects.	Implemented in .NET Remoting, Windows Communication Foundation.
Directory network infrastructure access	Implemented in Qt Network classes.	Implemented in Active Directory libraries.
Security, cryptography	Implemented in specially dedicated classes.	Implemented in .NET Framework Cryptography Model.
Messaging	Implemented in Qt Mobility	Implemented in Microsoft Message Query.
MVC implementation	Implemented in Qt modelling classes.	Implemented in ASP.NET MVC.
Multithreading	Implemented in Qt Concurrent, Qt thread pool and so on.	Implemented in .NET thread pool, .NET Parallel library and so on.

Advantage of .NET over Qt in this project is that modern version of .NET has built-in support of gaze tracking interface (through Windows 10 Gaze experimental API) (Microsoft Corp., 2019). Also, .NET is a native Windows technology, applications which were created using .NET don't demand to install additional runtime executables (.NET Common Runtime Environment is a part on any Windows version), while Qt is not installed by default on Windows operating systems and demands some additional actions from users (downloading and installing of appropriate versions of Qt dll files and other components). So, .NET Framework and C# programming language were chosen for this project.

6.1.4 Discussion on development technologies for gaze input

Gaze input is the relatively new away of communication between a user and a computer. This type of input demand presence of special device which is normally called eyes tracker. Eyes movement detection in this project relies on "Tobii Eye Tracker 4C" (of Tobii company). This is one of the such devices on market and its usage implies some peculiarities which entail from gaze input itself and used system level software (drivers and so on). The manufacturer of used device provides its own set of APIs which can be used in various type of applications for Windows operating system. Main target area of implementation of Tobii devices is modern versions of Windows (version 8 and 10).

In this project I implemented gaze input functionality using APIs which is provided for free by the manufacturer of the eye's tracker. This API contains collection of classes designed considering specific features of provided equipment (Tobii Group, 2018).

Functionality of the Tobii SDK API implements event-bus design pattern and uses built-in event architecture of .NET Framework. Events notify a client application that some important actions happened in the process of user's gazing. The most useful events are fixations and saccades. Also, possible to detect head movements and moments when a user appears or disappears.

Possible alternative API for this project is provided by Microsoft (Microsoft Corp., 2019). This API was introduced in Windows 10 April 2018 Update (Version 1803, build 17134). So, this is brand new feature of Windows operating system. It unifies requirements to eye tracing devices and allows different

Table 22: comparison Tobii and Microsoft gaze input technologies for this project

	Tobii	Microsoft
Supported frameworks	C++ runtime, .NET Framework.	.NET Core (Windows 10; Universal Windows Platform).
Supported devices	Tobii devices.	Any device which comply contract of the Gaze framework.
Fixation detection	Implemented	Implemented
Saccades detection	Implemented	No direct implementation
Head movement detection	Implemented	No direct implementation
Direct sending of gaze events to UI controls	Not implemented	Implemented

In general, Microsoft realization looks better because it supports wider range of devices. But concerning this project, this API demonstrated non-satisfactory efficiency of video rendering (because of unsuccessful update of its core software; obviously, this will be somehow corrected in the future, but I didn't have time to wait). Sensitive drawback of Microsoft API is narrow multitude of supported operating systems and its experimental status. Tobii SDK is a native SDK for the tracker being used in the project and provides full support of this device functionality. It implements many useful features, such as event handlers of gazing process, abstraction of interactors for getting information about head position, user's presence and so on. The only drawback of Tobii SDK entails from its benefits. Usage of this SDL makes my app less universal because it limits number of available devices. But anyway, my project is designed for concrete set of devices and not intended to be universal. This obstacle can be surmounted by on the level of realization of the app's architecture.

6.1.5 Discussion of development technologies for brain-computer interface

This project uses concrete EEG headset "NeuroSky MindWave Mobile 2 Headset" which is accompanied with native SDK provided by NeuroSky (NeuroSky Inc., 2018). By the moment there is not built-in support of brain computer interface in modern operating systems. So, the only option is to use native manufacturer's SDK. But, even in this case there some possible ways of programming of getting data from the BCI device being used.

First way is to get raw data of user's brain activity directly from the point of conjunction of EEG headset to the PC. Used headset is connected to the PC via wireless Bluetooth interface. In this case the program should find and connect to the registered paired Bluetooth device. After that it is possible to read stream of numeric data constantly being sent from the headset by using functions for interoperability with COM ports. This method has one serious disadvantage. Client application should decode packets of data according to guidelines provided by the manufacturer of headset. In this case there should be some criteria which allows to validate decoded data for correctness, consistency and adequateness. The problem is that I'm not a specialist in processing of brain data and cannot be sure that results of decoding of brain data could be correct. Because of that this way is not suitable. But it could be quite effective in the sense of efficiency and independency from third party software.

Second way is to rely on special professional middleware which is provided by NeuroSky. This software is permanently running server which processes data from registered paired headset and calculates cumulative analytical characteristics of a subject's brain activity, such as level of attention, level of relaxation, level of familiarity, level of mental effort, and values of standard brain waves (alpha, beta, gamma, delta, theta). This software is called "NeuroSky ThinkGear Connector". This apps sends data to local network port on the address "127.0.0.1:13854" over standard TCP/IP protocol. Data are sent in the form of JSON string representation (figure 24).

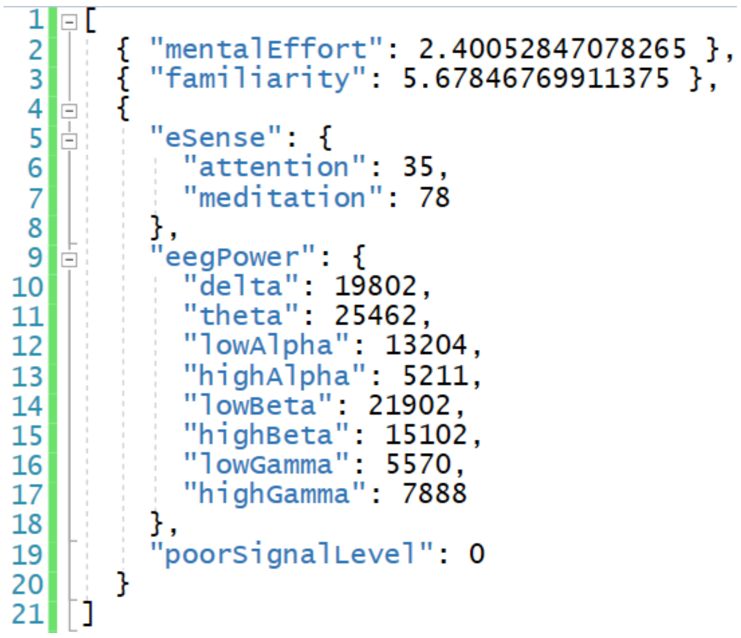

```
[
  { "mentalEffort": 2.40052847078265 },
  { "familiarity": 5.67846769911375 },
  {
    "eSense": {
      "attention": 35,
      "meditation": 78
    },
    "eegPower": {
      "delta": 19802,
      "theta": 25462,
      "lowAlpha": 13204,
      "highAlpha": 5211,
      "lowBeta": 21902,
      "highBeta": 15102,
      "lowGamma": 5570,
      "highGamma": 7888
    },
    "poorSignalLevel": 0
  }
]
```

Figure 24: Format of JSON data sent by ThinkGear server

So, in this case I can use standard classes of .NET Framework for programming of TCP connections and parsing of JSON data.

6.1.6 Discussion on realization of interoperability of PC and the robot

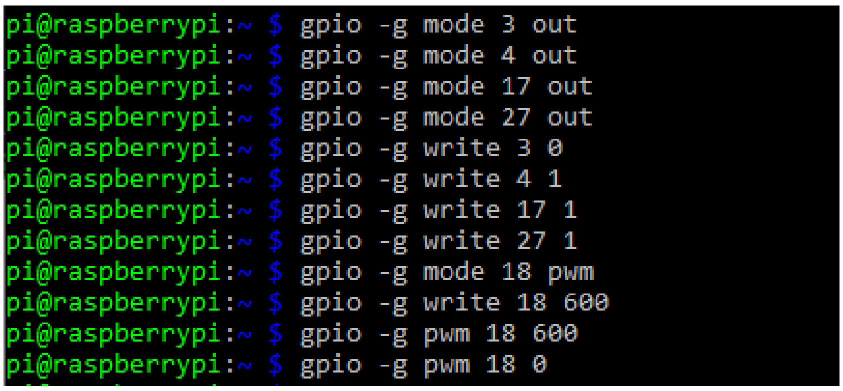
One of the major tasks of this project is to realize wireless interchange between the desktop app and the robot. The desktop app should be able to send motion commands to the robot in respond of the user's gaze motions. After receiving of the command, the robot should physically execute corresponding movements.

This functionality was implemented on the base of usage Raspberry PI GPIO interface (Warren, 2018). This interface is a hardware and software that allows to send physical electrical signals to the specially designated pins of the Raspberry PI computer. Each pin has its own address (number) and symbolic representation on the level of software implementation. Raspberry PI provides several ways to set values of the pins. When some device is connected to these pins it can be controlled by reading values of the corresponding pins and writing new values to the corresponding pin. Also, there is a built-in ability to control pins remotely.

Main problem of this project is to control the pins remotely. This can be achieved through usage of secure shell interface (SSH) (Dwivedi , 2003). This interface is implemented in all modern operating systems and can be used directly as CLI tool for remote control. Realization of SSH also realized both in Microsoft .NET Framework and Raspberry PI Linux operating system.

There are several ways to set values of pins. The most used is Python scripting (Nixon, 2015). It is possible to invoke Python scripts remotely using SSH. Major drawback of this approach is that it demands creation of complicated scripts for each type of movements with limited ability of changing inner parameters. Also, it is impossible dynamically rebuild sequence of command inside such scripts which could be possibly convenient for implementation of more flexible motion commands.

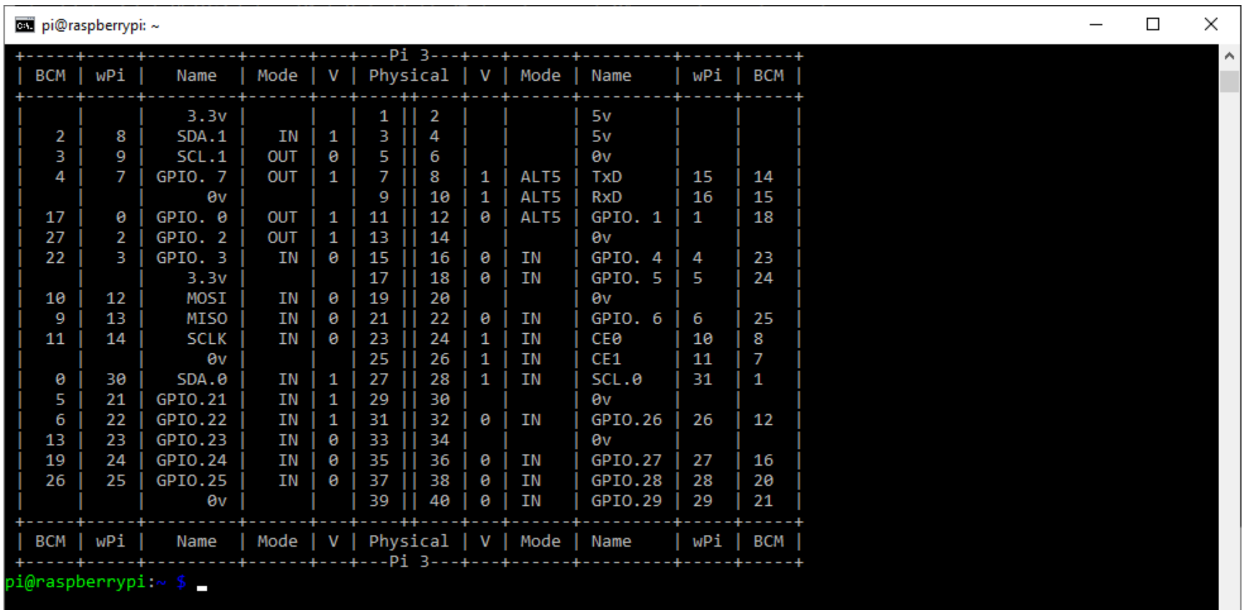
Instead of mentioned approach built-in CLI command of Raspberry PI bash was used. Raspberry PI has "gpio" command (Drogon, 2019). "gpio" (figure 25) is direct CLI command which can read and write pin's values individually (it can be invoked through SSH interface as shown below).



```
pi@raspberrypi:~ $ gpio -g mode 3 out
pi@raspberrypi:~ $ gpio -g mode 4 out
pi@raspberrypi:~ $ gpio -g mode 17 out
pi@raspberrypi:~ $ gpio -g mode 27 out
pi@raspberrypi:~ $ gpio -g write 3 0
pi@raspberrypi:~ $ gpio -g write 4 1
pi@raspberrypi:~ $ gpio -g write 17 1
pi@raspberrypi:~ $ gpio -g write 27 1
pi@raspberrypi:~ $ gpio -g mode 18 pwm
pi@raspberrypi:~ $ gpio -g write 18 600
pi@raspberrypi:~ $ gpio -g pwm 18 600
pi@raspberrypi:~ $ gpio -g pwm 18 0
```

Figure 25: Example of calls of "gpio" command in Windows ssh (command prompt)

This command can change mode of pins and execute many service commands (figure 26 shows command "gpio readall" which reports status of remote Raspberry PI pins in windows of Windows console (by using SSH interface)).



```
+-----+-----+---------+------+---+---Pi 3---+---+------+---------+-----+-----+
| BCM | wPi |   Name  | Mode | V | Physical | V | Mode | Name    | wPi | BCM |
+-----+-----+---------+------+---+----++----+---+------+---------+-----+-----+
|     |     |    3.3v |      |   |  1 || 2  |   |      | 5v      |     |     |
|   2 |   8 |   SDA.1 |   IN | 1 |  3 || 4  |   |      | 5v      |     |     |
|   3 |   9 |   SCL.1 |  OUT | 0 |  5 || 6  |   |      | 0v      |     |     |
|   4 |   7 | GPIO. 7 |  OUT | 1 |  7 || 8  | 1 | ALT5 | TxD     | 15  | 14  |
|     |     |      0v |      |   |  9 || 10 | 1 | ALT5 | RxD     | 16  | 15  |
|  17 |   0 | GPIO. 0 |  OUT | 1 | 11 || 12 | 0 | ALT5 | GPIO. 1 | 1   | 18  |
|  27 |   2 | GPIO. 2 |  OUT | 1 | 13 || 14 |   |      | 0v      |     |     |
|  22 |   3 | GPIO. 3 |   IN | 0 | 15 || 16 | 0 | IN   | GPIO. 4 | 4   | 23  |
|     |     |    3.3v |      |   | 17 || 18 | 0 | IN   | GPIO. 5 | 5   | 24  |
|  10 |  12 |    MOSI |   IN | 0 | 19 || 20 |   |      | 0v      |     |     |
|   9 |  13 |    MISO |   IN | 0 | 21 || 22 | 0 | IN   | GPIO. 6 | 6   | 25  |
|  11 |  14 |    SCLK |   IN | 0 | 23 || 24 | 1 | IN   | CE0     | 10  | 8   |
|     |     |      0v |      |   | 25 || 26 | 1 | IN   | CE1     | 11  | 7   |
|   0 |  30 |   SDA.0 |   IN | 1 | 27 || 28 | 1 | IN   | SCL.0   | 31  | 1   |
|   5 |  21 | GPIO.21 |   IN | 1 | 29 || 30 |   |      | 0v      |     |     |
|   6 |  22 | GPIO.22 |   IN | 1 | 31 || 32 | 0 | IN   | GPIO.26 | 26  | 12  |
|  13 |  23 | GPIO.23 |   IN | 0 | 33 || 34 |   |      | 0v      |     |     |
|  19 |  24 | GPIO.24 |   IN | 0 | 35 || 36 | 0 | IN   | GPIO.27 | 27  | 16  |
|  26 |  25 | GPIO.25 |   IN | 0 | 37 || 38 | 0 | IN   | GPIO.28 | 28  | 20  |
|     |     |      0v |      |   | 39 || 40 | 0 | IN   | GPIO.29 | 29  | 21  |
+-----+-----+---------+------+---+----++----+---+------+---------+-----+-----+
| BCM | wPi |   Name  | Mode | V | Physical | V | Mode | Name    | wPi | BCM |
+-----+-----+---------+------+---+---Pi 3---+---+------+---------+-----+-----+
pi@raspberrypi:~ $ 
```

Figure 26: Output of "gpio readall" command in the console of Windows

This command can be invoked remotely through SSH in the manner of usual CLI command is normally invoked. This allows to create packages of motions commands on the side of the desktop app and send them to the robot. These packages of commands can be formed dynamically, and their inner structure can be altered easily accordingly to the aims of the app. The last-mentioned circumstance made "gpio" CLI utility ideal for this project as a tool for execution of motion commands of the robot.

6.2 Dependencies of software components of the project

For further explanation of the software architecture and structure of its components it would be useful to describe overall structure of dependencies between software components which are involved in this software and hardware system's functioning (figure 27). Further explanation in section 7 concerns "Gaze and Mind Driver Application" and "Video capture Web-app".

Two programs were created to fulfill the project aims. First is "Gaze and Mind Driver" which executes all the functions for receiving data from external devices, shows video stream from the robot, forms and send motions commands to the robot. Second is a script for implementation of interaction between the robot (and its video camera) and mentioned application.

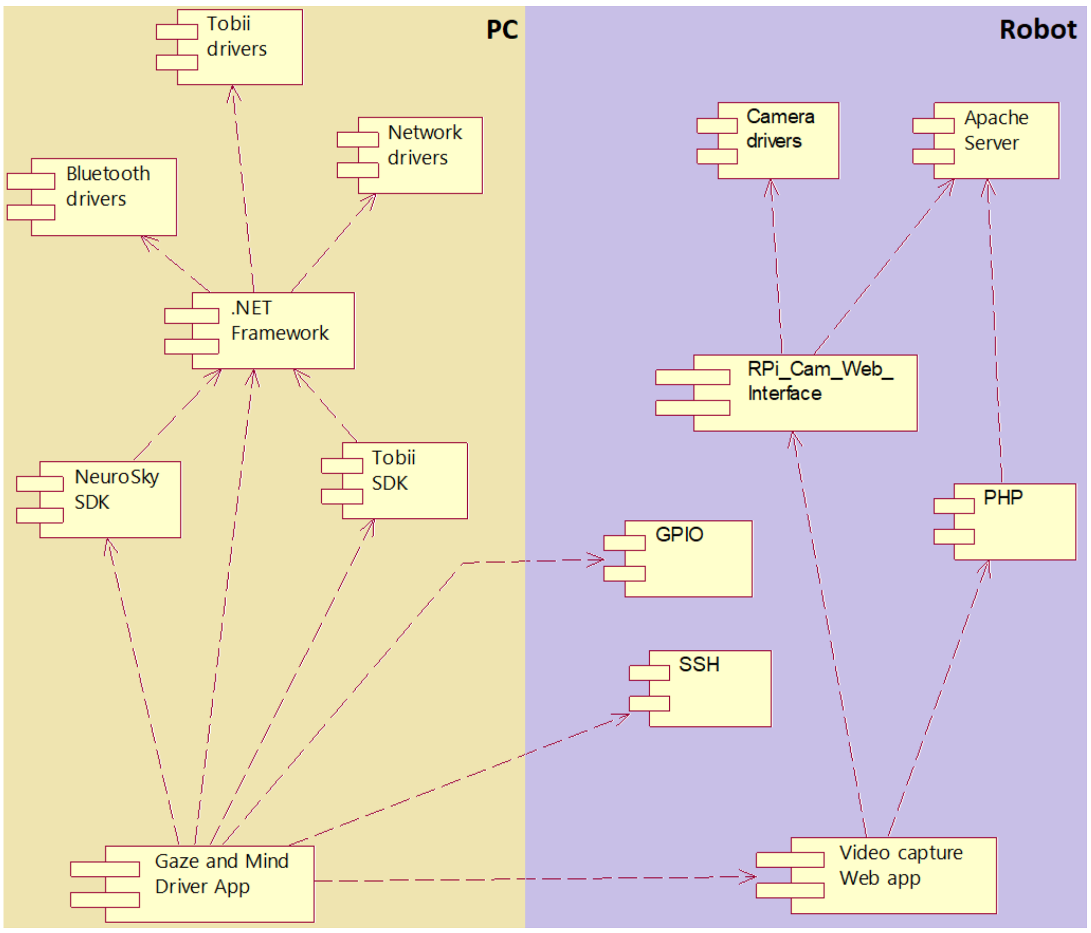


Figure 27: Software components of the project (UML component diagram)

6.3 Reflection

During the process of hardware design and choice of basic software and technologies I encountered some problems which reflect some ambiguous nature of the system being developed. This system consists of two principal parts which interact and influence each other. It was not clear from what part of the system I needed to start the development.

Overall functionality and effectiveness of this system depend on not only from my personal efforts as a software developer but also from used third-party components such as physical devices (their effectiveness almost is out of my control and I need do adapt software being created to the equipment's peculiarities) and low-level software components which were provided by manufacturers of the devices. This type of development was relatively new for me and demanded thorough thinking and delving inside hardware part of the project. I and the project coordinator spent some time on assembling the robot and connecting its electrical parts together in correct order. Also, we spent some time on forcing this system to work as was expected. This process allowed me to get valuable experience because I had never implemented hardware systems from the scratch before. This process gave me understanding what components of software should be created and how they should behave. On the base of these activities I managed to approach the choice of software components consciously and I hope that this chapter adequately describes all major circumstances of that process.

7 SOFTWARE IMPLEMENTATION

7.1 Architectural patterns

While planning and designing the main desktop app and the robot's server-side software I tried to keep following standard recommended practices of software development. The aim was to create robust and reliable system which meets the requirements and allows to use gaze input and EEG data to control motions of the robot.

First of all, I would like to repeat that overall system's architecture follows architectural peculiarities of software for robotic systems (section 5.1.6). I used three-layered software architecture (planner-executive-functional) for robotic systems (Ahmad & Ali Babar, 2016) which allows to build decoupled system in the sense that software components of each layer are not strongly dependant from each other. Each level has some sort of autonomy in logical sense.

Whether the program is designed for robot's control or some other tasks it should qualify standard demands and recommendations in order to guarantee that quality characteristics are met.

According to quality assurance plan (section 3.7, table 7) I need to achieve many quality characteristics. Usage of architectural patterns can serve this goal (Cervantes & Kazman, 2016; Flowler, 2013; Mallawaarachchi, 2018).

7.1.1 Discussion on usage of client-server architectural pattern

Client-server architecture is used to implement sending and receiving of video stream produced by robot's camera. Video stream is needed to inform the user about current position of the robot. This video stream is used as a means of implementation two-way back connection. I mean that picture influences the user's decision to move robot to some direction and at the same time actions of the user change the picture (because position of the robot is changed).

There are following correspondence between structure of my project and this pattern (Figure 28).

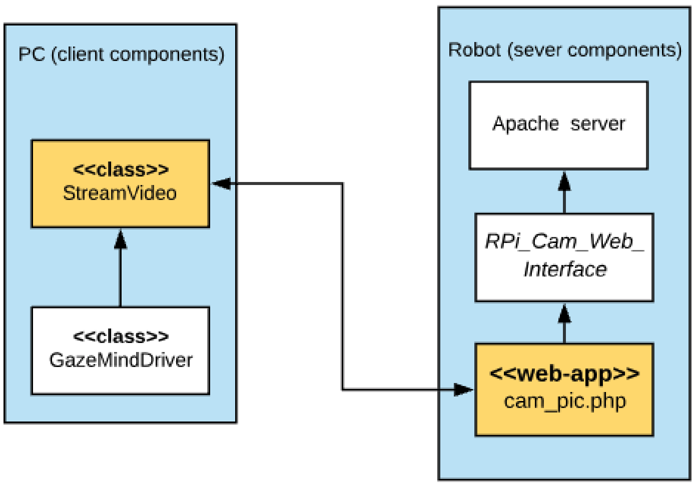


Figure 28: Schematic representation of "client-server" architectural pattern in my project

Usage of this pattern allows to implement direct communication between desktop app and the robot. Benefits of this architecture include such effect as scalability and expandability. This means that I could improve and expand this project by using some new types of hardware (for example, infrared sensors for avoidance of collisions with obstacles). In this case additional module can be implemented on the sever side. Also, this approach allows to use standard reliable components such Apace server to provide some customized functionality. Working with this aspect of the system was relatively new for me because server part of the system had to implement interchange with video data. Main problem was not in architecture itself. Efficiency and speed rate of generated video stream was not stable and depended of used middleware of the server side (in my project I used RPi_Cam_Web_Interface component).

7.1.2 Discussion on usage of event-bus architectural pattern

This application is significantly relied on idea of event-bus architecture. I implemented my own classes which can raise custom events connected with actions of user's gaze input and user's brain activity. Events implement important functionality which allows to inform the app that gaze fixation or saccade happened, or level of user's brain relaxation level became too high. At the same time this idea gives my project such positive feature like loose coupling of classes. An event is normally accompanied by special object which can carry information about this event (classes which exchange events don't need to know about implementation details of each other). In my project I implemented classes with events as shown in figures ***. Here I need to mention that the role of the bus component is played by .NET Framework Common Language Runtime (CLR).

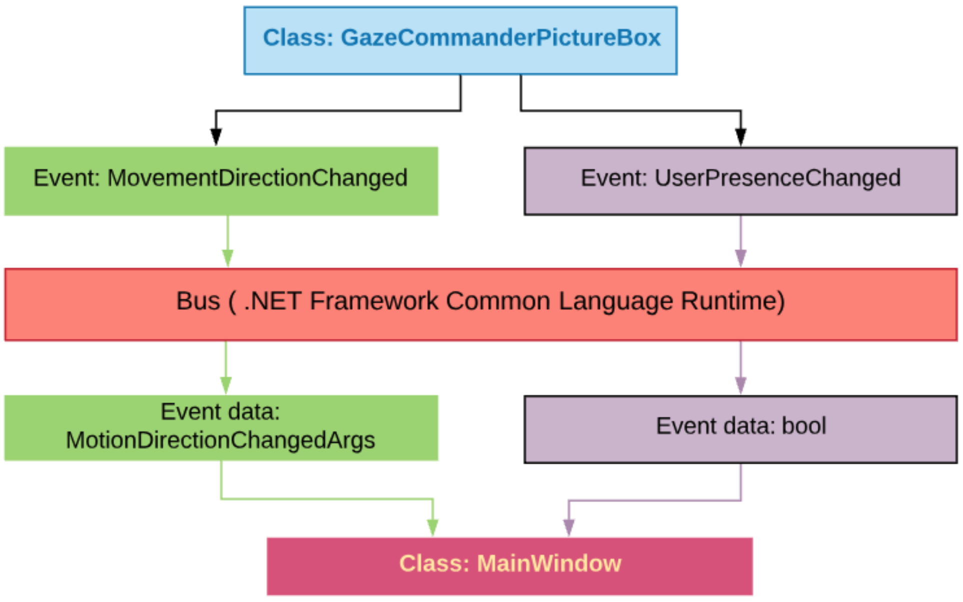

Class: GazeCommanderPictureBox
Event: MovementDirectionChanged
Event: UserPresenceChanged
Bus (.NET Framework Common Language Runtime)
Event data: MotionDirectionChangedArgs
Event data: bool
Class: MainWindow

Figure 29: Schematic representation of "event-bus" architectural pattern in my project

Here such classes as MainWindow and GazeCommandPictureBox don't "know" about realization of their inner mechanism. But because of presence of event with their concomitant data, which is delivered by .NET CLR, *MainWindow* can receive information when direction of user's gaze was changed or when the user left the computer or when the user appeared in front of the computer again. Changes in GazeCommandPictrueBox, which shows video stream and receives gaze input don't demand any special changes in main window of the app (which contains and shows this component). Data of events is carried by appropriate event objects MovementDirectionChanged and UserPresenceChanged (they are instances of event type of C# programming language). The same idea is implemented for EEG headset interoperability (figure 30).

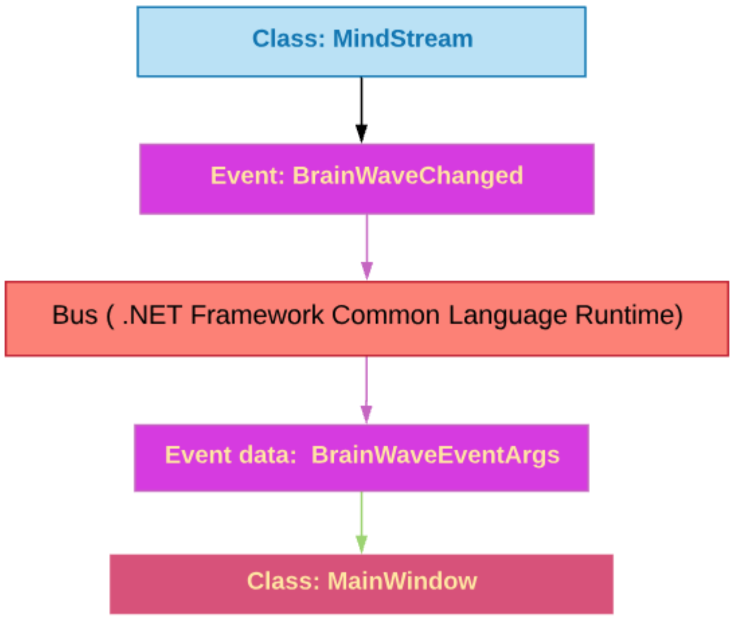

Class: MindStream
Event: BrainWaveChanged
Bus (.NET Framework Common Language Runtime)
Event data: BrainWaveEventArgs
Class: MainWindow

Figure 30: Schematic representation of "event-bus" architectural pattern in my project

.NET Framework delivers quite convenient way to adapt "event-bus" architectural pattern in the applications. This adaptation provides my project with set of loosely coupled (data coupled) (Booch, 1991) major classes which act independently for achieving of common goal of forming and sending commands to the robot.

In the case of this project this pattern also could deliver some problems of overall efficiency of the app. The gist of the problem that devices being used provide very tight stream of events because have rather high level of sensitivity and rise events on any change (of gaze position or of EEG waves). This can create unnecessary overloading of processor and possible unresponsive UI. Firstly, this problem was solved by implementing of multithreaded architecture of the app. Secondly, filtration of data from devices was used in order to handle only significant events which signal about important changes of state of observed data. Not every gaze change should be converted to motion command but only that which move controlling cursor to specific area inside main window of the app. Only when brain activity fells down below the predefined value there is a sense to send event about changes in the user's brain activity.

7.2 Design patterns and other details of implementation

7.2.1 Discussion on usage of "adapter" pattern

According to standard definitions and descriptions (Gamma, Helm, Jonson, & Vlissides, 1994) this pattern (adapter) is used when there is a need to convert data of some class (let's call it A) for usage in other class (B) which can't directly use data produced by class A. In my project usage of this pattern contributes to the principle of separation of concerns. I have class which handles gaze input and other class which produces commands for the robots. Gaze input provided being caught is presented in the form of information about concrete specific region of the UI where user's gaze was registered. This data is needed to be converted to the actions of the robot. Classes which create commands can't operate and use data about regions of UI of the app because this is not their zone of responsibility. Converting of data from shapes of UI region's description to invocation of executable commands is made by adapter class.

Class GazeRectangleMotionAdapter allows to retrieve reference to a method which corresponds to the area where gaze point was lastly caught. When this reference is retrieved (on the base of inner conversion of data to method (or delegate of C# programming language)) it can be used for invocation of referenced method which force the robot to move.

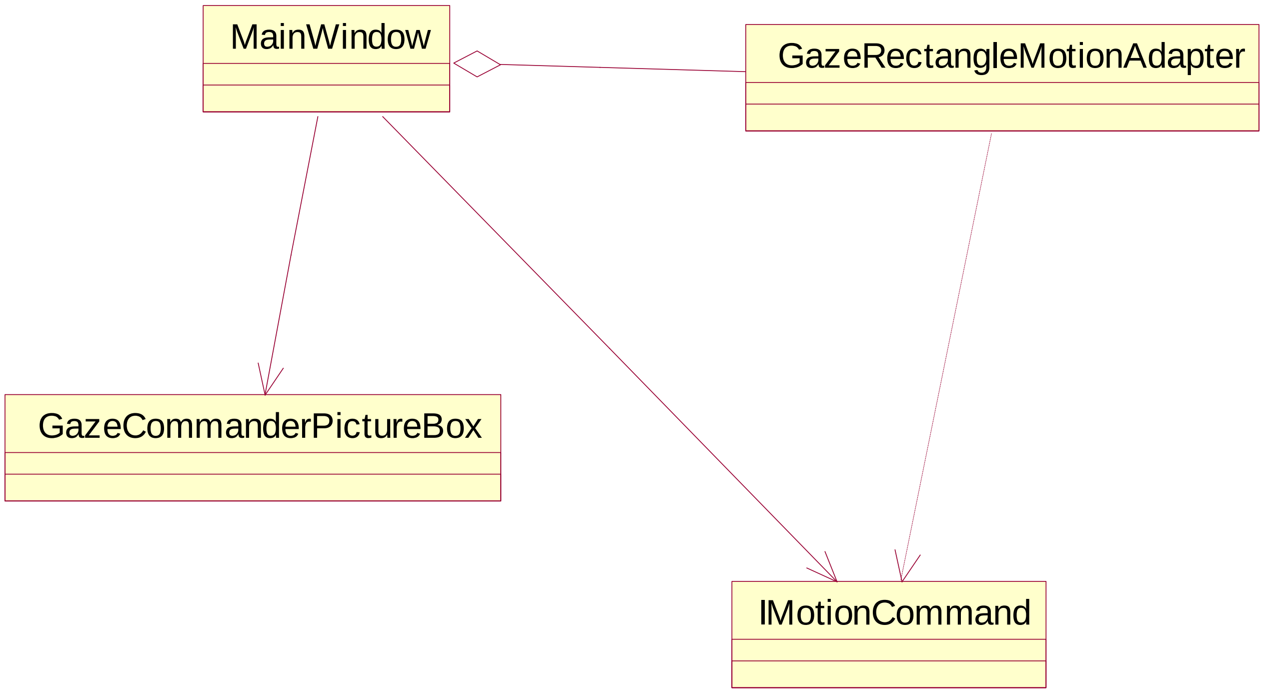


Figure 31: Implementation of "adapter" design pattern in my project (UML class diagram)

Sense of this diagram can be described as follows. The object of MainWindow class gets data about user's gazing from the object GazeCommandPictureBox and uses instance of GazeRectangleMotionAdapter class (inner class of MainWindow) to retrieve reference to a method of GPIOSshMotion class (through reference to the interface IMotionCommand).

I would say that some slight negative effect is that structure of the project could look quite complicated because there is no straightforward indication of how data is transformed to commands.

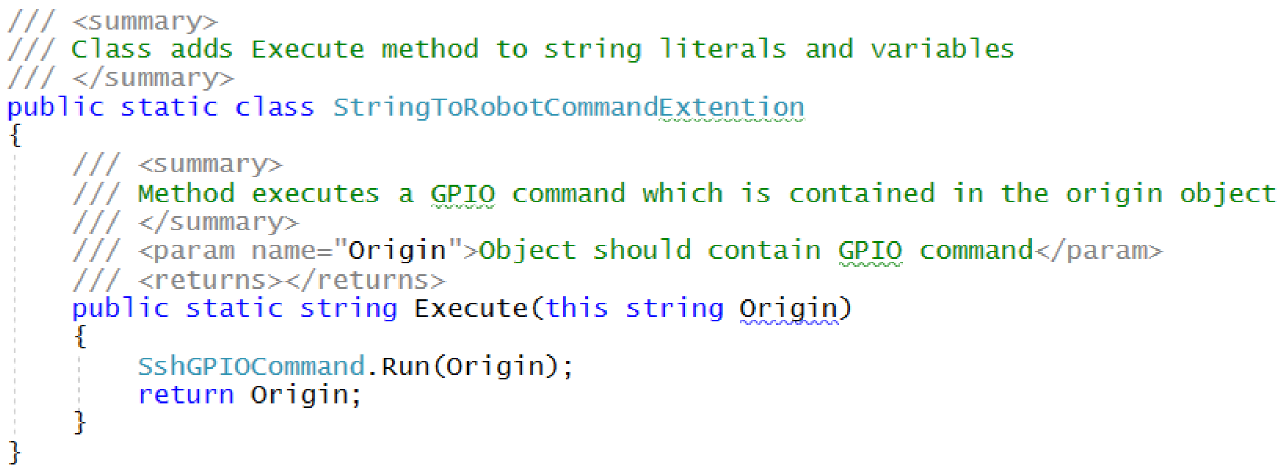
7.2.2 Discussion on usage of "visitor" pattern

Authors of used sources (Gamma, Helm, Jonson, & Vlissides, 1994) describe "visitor" pattern as a means that allows to implement some operation on the object of some class when this operation doesn't pertain to this class. In other words, this pattern is used when we have some class which doesn't have some needed functionality, and we need to add this functionality to this class without changes in its inner structure (because it can be forbidden or undesirable).

If to say about my project, I faced such a necessity to implement additional functions (methods) in standard classes of .NET Framework (String, Point). These classes are not available for inheritance and thus I could not simply take their initial definition and create derived classes with needed functionality. Visitor pattern which was used in my project allows to follow "opened-closed" part of SOLID (Booch, 1991) principles. According to this principle classes should be opened for extension and closed for modifications.

Implementation of this pattern was made by adapting of such feature of C# programming language as extension. This syntax construction allows a programmer to add new methods to existing classes without any changes in the structure of these classes.

For example, I defined new class which is called *StringToRobotCommandExtention*. This class satisfies definition of visitor pattern and extends String class of .NET Framework.



```
/// <summary>
/// Class adds Execute method to string literals and variables
/// </summary>
public static class StringToRobotCommandExtention
{
    /// <summary>
    /// Method executes a GPIO command which is contained in the origin object
    /// </summary>
    /// <param name="Origin">Object should contain GPIO command</param>
    /// <returns></returns>
    public static string Execute(this string Origin)
    {
        SshGPIOCommand.Run(Origin);
        return Origin;
    }
}
```

This static class has quite simple implementation. It consists of one static method which has String parameter. Idea of this method is to add method Execute to string literals and variables (in the scope where this extension is used). Module which implements motion commands needs to send command to the robot. Each such command eventually is a string of the following format (CLI SSH GPIO command): "gpio -arguments". Created extension allows to write such commands in the code of the app like follows:

```
"gpio -g write 18 600".Execute();
```

or

```
String command = "gpio -g write 18 600";
command.Execute();
```

This style of command invocation looks more natural and more straightforward. This method extends String class and doesn't change it. For that reason, this class meets definition of "visitor" pattern. Here I should say about main drawback of implementation itself (which doesn't make used pattern worse). This implementation needs to check source string for semantical correctness (origin string should contain correct GPIO command). But in the case if origin string contains not correct command server side simply ignores it and doesn't send this command to the robot.

In some sense used approach (I mean usage of the "visitor" pattern) could confuse a reader of the source code to the app. Mentioned circumstance describes some disadvantage of "visitor" pattern's implementation I used here. I decided to use it because it makes code more concise and intuitive. But extensions in C# are not absolutely new and their usage can be considered quite justified.

7.2.4 Discussion on usage of interfaces

I tried to follow the best recommended practices of object-oriented design during all the phases of this project. In particular, I mean SOLID principles. One of them is Liskov substitution principle which states that usage of some class should be allowed to be substituted by any offspring of this class (McLean, 2014). Concerning my project, following this principle can contribute such quality characteristic as compatibility with wider range of devices of gaze input. This can reduce needed time for modification of my code in the case when a class of some new device should be introduced. In other words, it increases maintainability and extendibility of the app. I implemented this principle by utilizing of interfaces (Figure 32).

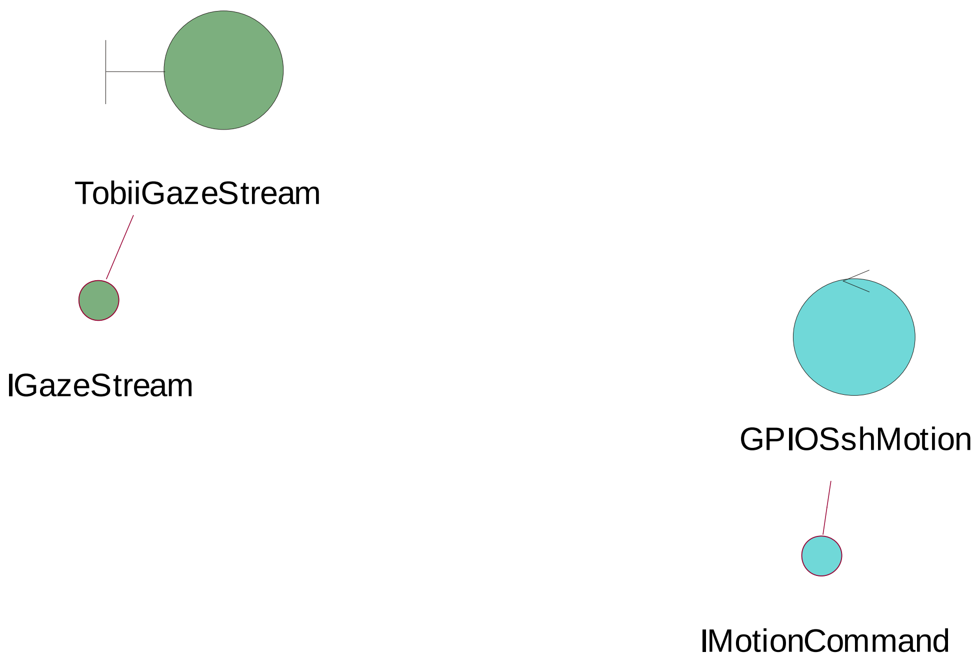
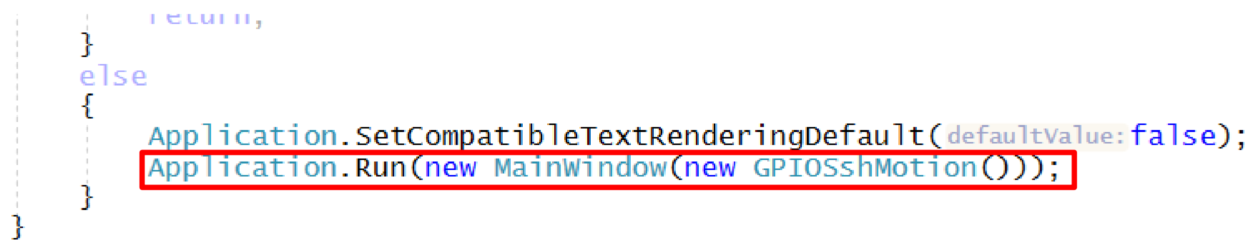


Figure 32: Fragment of UML class diagram shows implemented interfaces

Interfaces in my project model common behavior of gaze devices of different types and motion commands with possible different implementation.

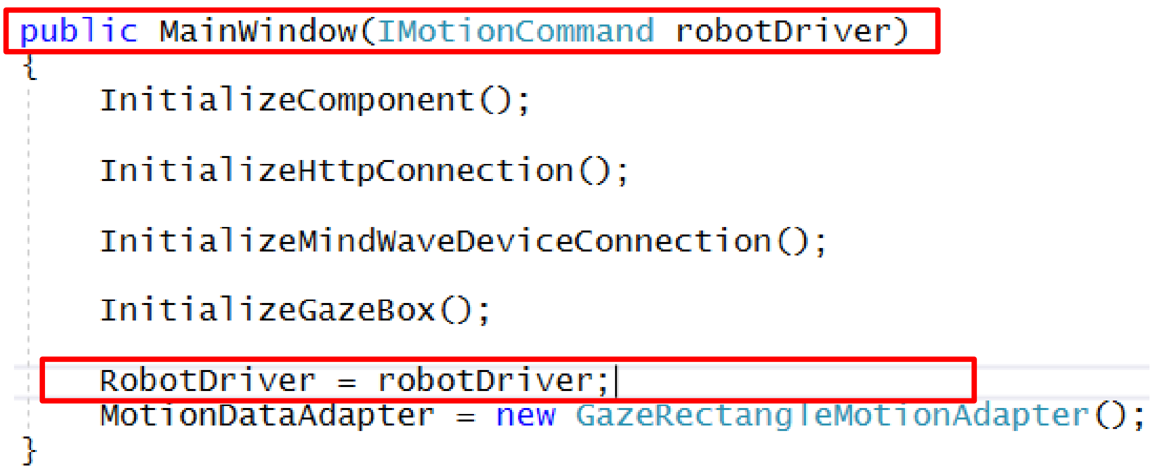
I declared interface IMotionCommand and implemented this interface in GPIOSShMotion class. This architecture potentially allows to introduce new types of motions (this could be motions of different type of motors or this motion can be implemented through other types of low-level interfaces) without necessity of make changes in MainWindow class.

For example, MainWindow class is created as follows.



```
            return;
        }
        else
        {
            Application.SetCompatibleTextRenderingDefault(defaultValue:false);
            Application.Run(new MainWindow(new GPIOSshMotion()));
        }
    }
```

Constructor of the MainWindow class is implemented as follows.



```
public MainWindow(IMotionCommand robotDriver)
{
    InitializeComponent();

    InitializeHttpConnection();

    InitializeMindWaveDeviceConnection();

    InitializeGazeBox();

    RobotDriver = robotDriver;
    MotionDataAdapter = new GazeRectangleMotionAdapter();
}
```

This means that I could instantiate robot's driver (which serves for sending commands to the robot) of MainWindow class by any object which belongs to some class implementing universal interface IMotionCommand. In this case, changes in used motion's class and command of creation of MainWindow's object don't lead to any changes in definition of MainWinow class.

Usage of this implementation allows to follow interface segregation principle and to upgrade level of code reusability. This approach was adapted not for the scratch but by the end of the development process. I would say that namely this project doesn't need such implementation. But this system is to be a prototype of systems which is potentially could be driven by different types of eyes trackers and could be moved by different types of motors. Because of that I've decided to implement idea of Liskov substitution principle which can give better scalability and adaptability of created software.

7.3 Multithreading and inter-process communication

7.1.1 Discussion on usage of multithreading

I've already mentioned that peculiarities of used equipment make overall efficiency vulnerable because of tough stream of data and events from used devices. This fact could be understood if to say that EEG headset works with quite high baud rate and provide the system with information about the user's brain data with frequency of several tens of data packets per second. The same can be said about used eye tracker. This device catches any saccade and fixation of user's eyes and immediately reports to the system. Considering that human eyes are constantly moving its easy to imagine intensity of data stream. Practical experiments showed that usage of usual one-thread architecture leads to unresponsive behavior of the app when delays of sending of commands can happen or the main window fails to show real-time video stream from the camera. The most effective way to solve such a problem is to use several independent threads of execution (Cleary, 2014). Each thread should be responsible for receiving and filtering of data and events from used devices.

My application has up to five independent parallel threads of execution which correspond to work of main method, window UI, EEG data processing, and execution of the robot's motion command (eye tracking was implemented of the UI thread and fully based on event-bus architecture because needs quick access to the main window of the app for rendering position of gaze cursor). First of all, I would like to describe overall behavior of the system.

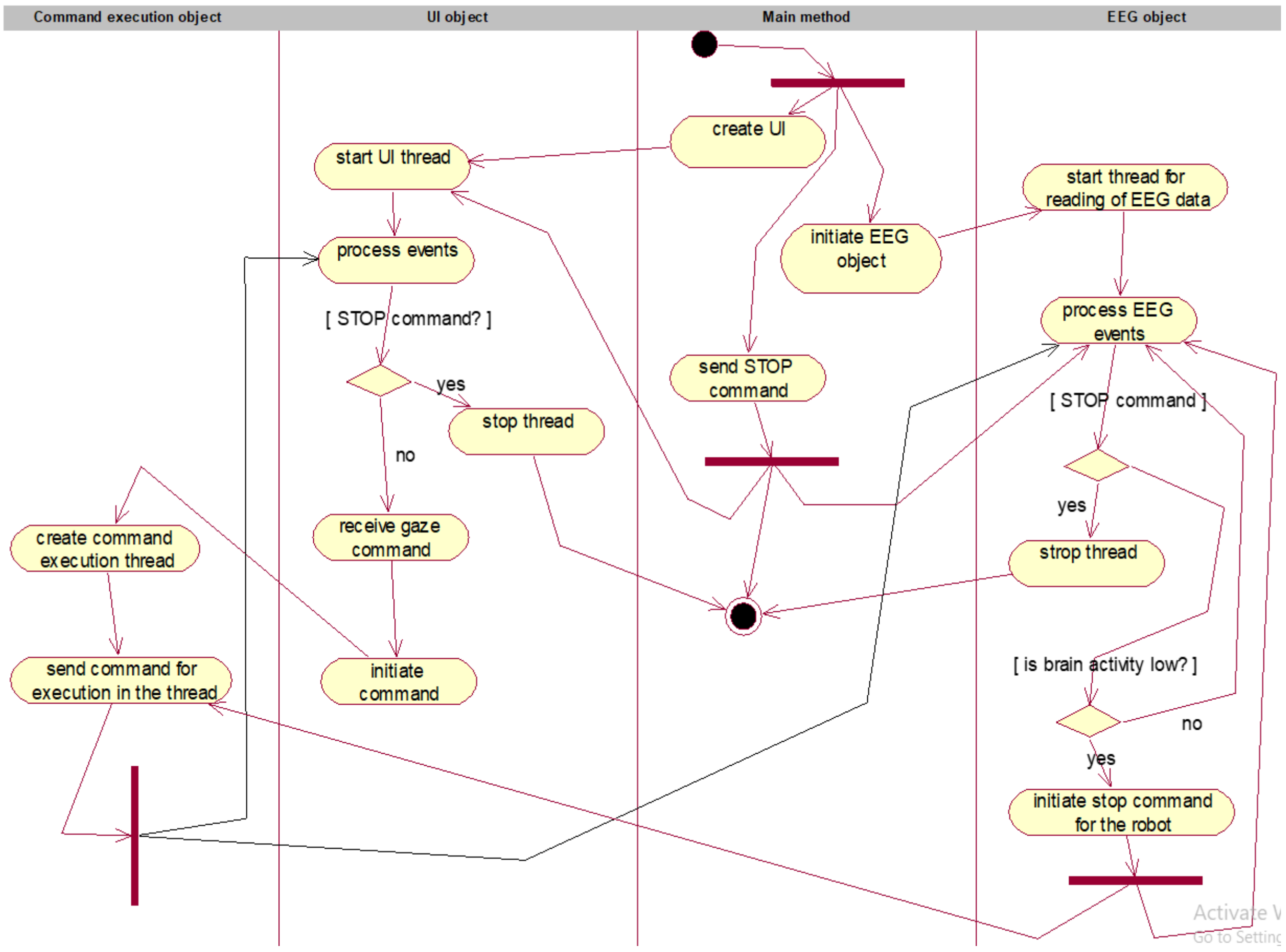


Figure 33: Schematic representation of implemented multithreaded (UML activity diagram)

Additional thread for receiving and showing of video stream is not shown (this thread is implemented by using the functionality of system timer because video stream can look smoothly if being refreshed with the frequency not less 25 times per second).

First lane of the diagram shows that this subsystem can create new threads quite frequently for any new command. It is needed because each motion command is sent over relatively slow network channel (if to compare this channel with speed of inner app commands). First implementations of motion commands were not efficient because any invocation made the app slow and unresponsive. When I realized the gist of the problem, I decided to put invocation of any motion command on separate thread. After that execution of motion command became more efficient and UI and overall performance stopped to suffer from hangings.

But any thread is a consumer of processor's time slices and system memory. Also, creation of a thread takes some time. My app creates many threads, and this itself can become a source of slowdown. This can simply kill benefits of used multithreading. This negative effect can be mitigated by usage of preliminary created threads from .NET Framework thread pool. This is a system level feature of .NET and allows get a new thread by request from the app. I used thread pool for quick retrieving of threads for my purposes. Profiling of the app showed that the app's performance increased when I started created threads using thread pool (I used Task class which can retrieve new thread in asynchronous manner).

Another problem which arises from the multithreading relates to usage of shared data. It is common problem of multithreaded apps which well described in the different sources (Cleary, 2014). My implementation forbids usage of mutable shared data which allows to avoid concomitant problems (such deadlock, starvation, races and so on). If threads of my app need to use some common data, then they are allowed to read it from specially created common static immutable classes of the app.

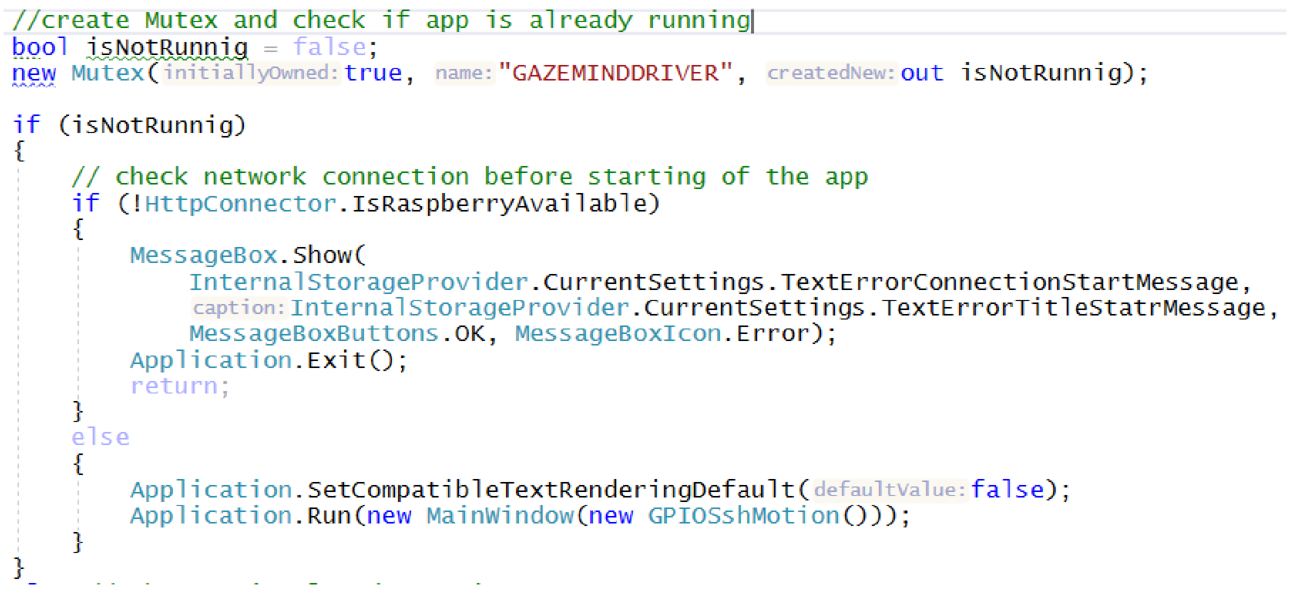
Undertaken profiling showed that this implementation reduced consumption of the processor time, but slightly increased consumption of system's memory (because each thread demands memory resources). This also allowed to get rid of unresponsiveness of the UI and reduce latency of video streaming in the main window of the app.

7.3.2 Discussion on usage of objects of inter-processes synchronization

In this project I had to implement such a behaviour which excludes launching of more than one copy of my app within one computer. This is obvious idea because the system has only one robot and controlling it from different copies of the same app can to lead to controversial commands and unstable state.

This task was solved by the usage of an object of inter-process communication. Any instance of launched app is a process of the operating system. Windows operating system allows one process to create special objects which are reachable from other processes. Such objects are stored in specially dedicated section of operational memory of operating system and controlled both by processes and the operating system.

In this project I used mutex (mutual exclusion object) which has its own unique name within the operating system memory. .NET Framework provides its own implementation of the mutexes in the class named Mutex. While new instance of the app is launching it is checked whether this mutex (with concrete name) exists or not. If mutex with this name doesn't exist, then the app creates new named mutex and continues standard launching process. If the mutex with this name already exists, then process of launching is stopped by the algorithm of the app and control flow leaves this new instance of the app. That means that new instances of the app cannot be created. When the app finished then operating system removes all resources which relate to its process including its mutex.



```
//create Mutex and check if app is already running
bool isNotRunnig = false;
new Mutex(initiallyOwned:true, name:"GAZEMINDDRIVER", createdNew:out isNotRunnig);

if (isNotRunnig)
{
    // check network connection before starting of the app
    if (!HttpConnector.IsRaspberryAvailable)
    {
        MessageBox.Show(
            InternalStorageProvider.CurrentSettings.TextErrorConnectionStartMessage,
            caption:InternalStorageProvider.CurrentSettings.TextErrorTitleStatrMessage,
            MessageBoxButtons.OK, MessageBoxIcon.Error);
        Application.Exit();
        return;
    }
    else
    {
        Application.SetCompatibleTextRenderingDefault(defaultValue:false);
        Application.Run(new MainWindow(new GPIOSshMotion()));
    }
}
```

Ideally, this problem should be treated more globally. The problem is that the system has only one robot and it should not be controlled from different computers; or probably it would be good idea to implement control from one main computer and some other specially allowed computer in the cases of some emergency. This means that the app should work in the only instance in the internet. This idea is not implemented in my project. But potentially it could be realized by usage of global shared state object which is needed to be placed on specially dedicated cloud service or web-server. Another way is to count number of connections to the robot with its inner software component.

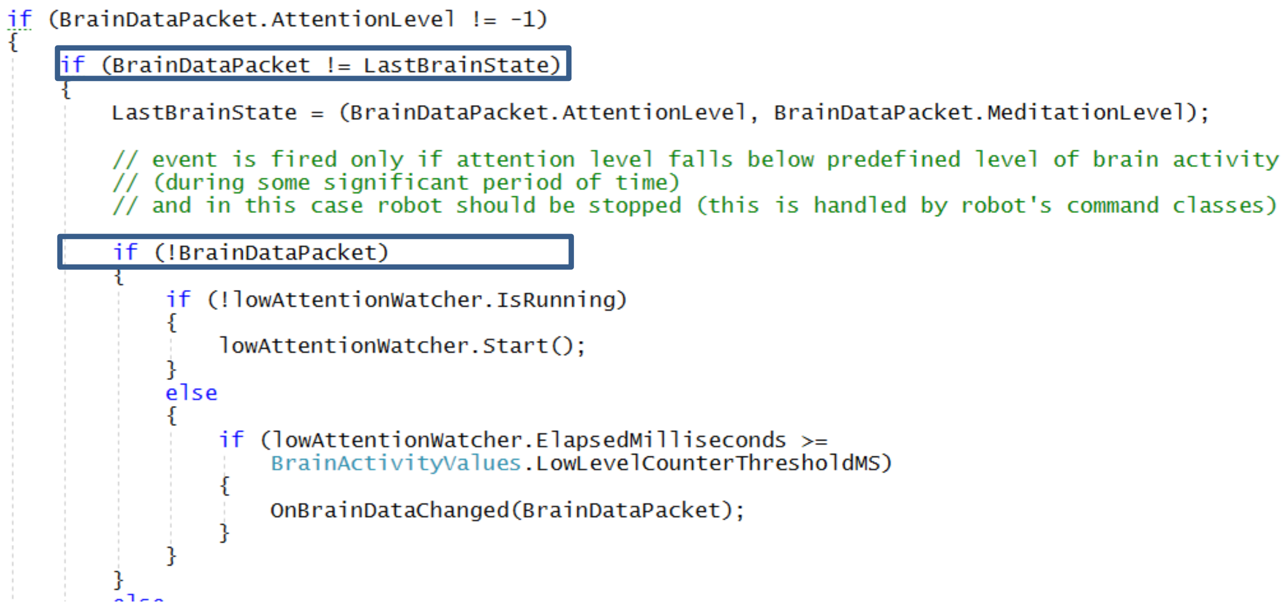
7.4 Special features of C# and .NET used in the desktop app

7.4.1 Discussion on usage of operators overloading

Operators overloading (Troelsen, 2017) is an arguable feature which is used in some object-oriented languages (C++, C#, Swift) and strictly rejected by others (Java). I used this feature to implement some syntactic simplification of my code. Expediency of this overloading can be justified by relatively transparent meaning of the operation which I overloaded. I suppose used overloading contributed to more clear code which could deliver better readability and maintainability of the app.

```
public static bool operator == (BrainWaveEventArgs Left, (int Att, int Med) Right)
public static bool operator != (BrainWaveEventArgs Left, (int Att, int Med) Right)
public static bool operator ! (BrainWaveEventArgs Operand)
```

Overloaded versions of ==, !=, ! operators were implemented for class BrainWaveEventArgs. The main idea is to simplify statements which verify level of brain activity of the user. Operator "!" means the negation of current state of the user's mind. This means that if level of attention and relaxation are too low then robot should be stopped if it's running at this moment. Usage of new implementation of this operator is shown below.



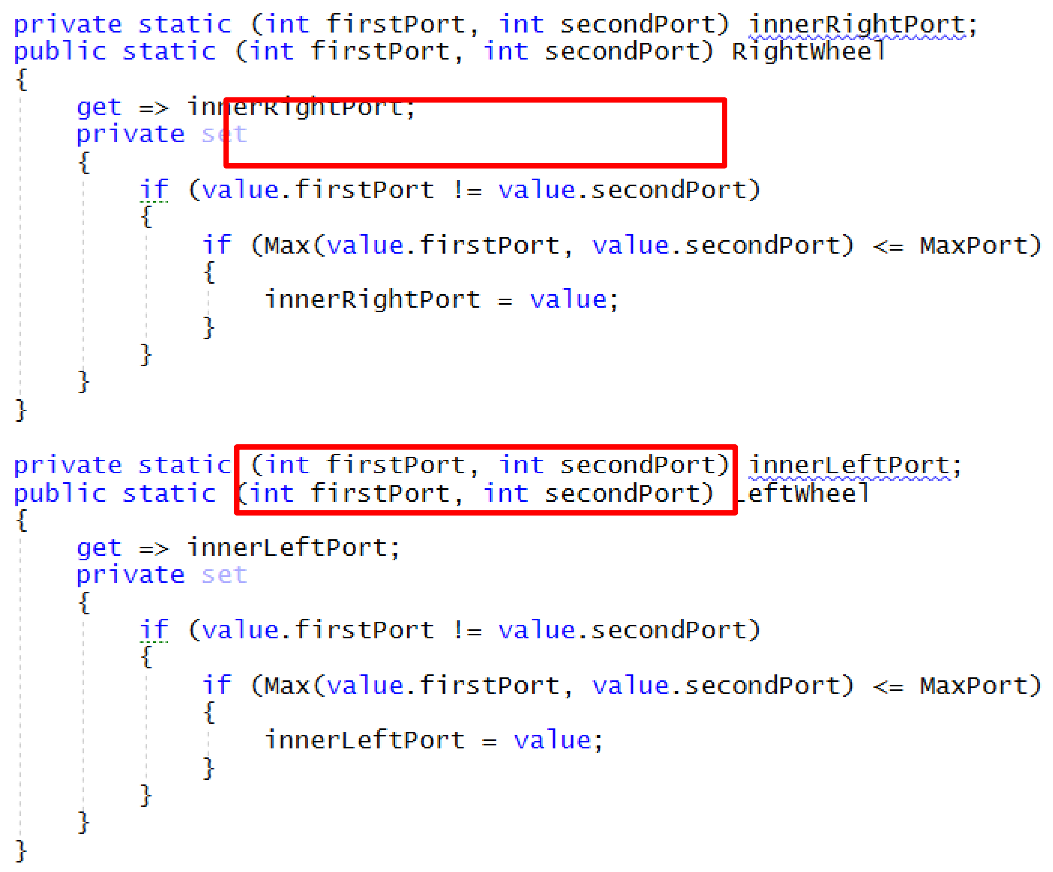
```
if (BrainDataPacket.AttentionLevel != -1)
{
    if (BrainDataPacket != LastBrainState)
    {
        LastBrainState = (BrainDataPacket.AttentionLevel, BrainDataPacket.MeditationLevel);

        // event is fired only if attention level falls below predefined level of brain activity
        // (during some significant period of time)
        // and in this case robot should be stopped (this is handled by robot's command classes)

        if (!BrainDataPacket)
        {
            if (!lowAttentionWatcher.IsRunning)
            {
                lowAttentionWatcher.Start();
            }
            else
            {
                if (lowAttentionWatcher.ElapsedMilliseconds >=
                    BrainActivityValues.LowLevelCounterThresholdMS)
                {
                    OnBrainDataChanged(BrainDataPacket);
                }
            }
        }
        else
```

Interestingly to notice that this feature is considered by different specialists as benefit (Troelsen, 2017) and disadvantage (Muhlestein, 2018) at the same time. The main reason for usage of this kind of overloading is that I can my code more concise and cleaner. Also, this overloading doesn't affect the performance (Troelsen, 2017). Possible alternative of this approach is usage of additional method with standard notation (with custom name and list of parameters). This version has such positive side as more standard view of invocation. But anyway, real understanding of what is made by the command doesn't depend on the form of its invocation and demands reading of source code. Because of that, classical representation of usage of class' elements (without implementing of operators overloading) doesn't have significant advantages. So, I suppose that usage of operators overloading in my project can be considered as expedient.

7.4.2 Discussion on usage of tuples

Tuples are not new feature in programming languages, but they are not widely used. C# programming language became tuples-friendly since the pre-last version. I used tuples for this project in order to define used pairs of pins of GPIO of Raspberry PI. Each wheel of the robot is controlled by two pins of GPRIO. Concrete value should be set to both of the pins in order to force wheel rotating forward, backward or to stop.

According to used hardware architecture, left wheel is controlled by pins with numbers 3 and 4 and right wheel is controlled through pins with numbers 17 and 27. It would be convenient to manipulate not with 4 different values but with two pairs of values because they can represent namely two wheels of the robot. Implementation is shown below. I believe that usage of tuples can improve logical structure of the code because tuples of two integer values can more precisely describe control mechanism of the robot wheel. In other word they provide visible abstraction of programmed entity. Possible alternative of this approach can be creation of a class for modelling data of a wheel. I would say that tuple has obvious advantage over a class because of its inline style. Tuple is defined in the place of usage and is visible explicitly. Usage of a class to store two integer values could be considered as overkill and doesn't provide needed readability of the code. Possible drawback is that tuples are new for C# language and are not supported by previous versions of .NET Framework.



```
private static (int firstPort, int secondPort) innerRightPort;
public static (int firstPort, int secondPort) RightWheel
{
    get => innerRightPort;
    private set
    {
        if (value.firstPort != value.secondPort)
        {
            if (Max(value.firstPort, value.secondPort) <= MaxPort)
            {
                innerRightPort = value;
            }
        }
    }
}

private static (int firstPort, int secondPort) innerLeftPort;
public static (int firstPort, int secondPort) LeftWheel
{
    get => innerLeftPort;
    private set
    {
        if (value.firstPort != value.secondPort)
        {
            if (Max(value.firstPort, value.secondPort) <= MaxPort)
            {
                innerLeftPort = value;
            }
        }
    }
}
```

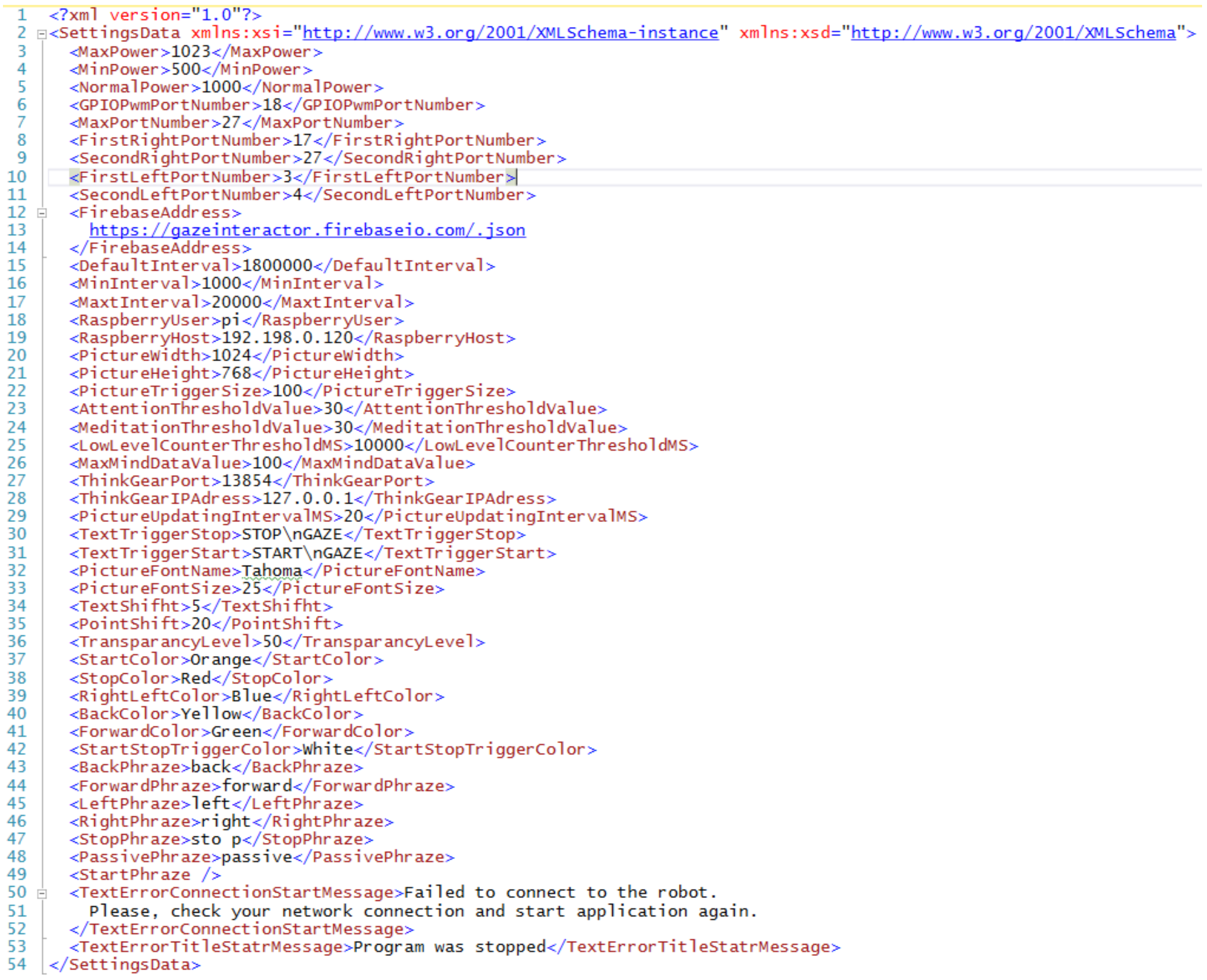
7.5 Discussion on data storage implementation

According to the aims of the project and requirements (section 5.2, table 15, SRF-16, 17, 18), software being developed should implement storage of its inner settings in the structured file of XML format.

XML format was chosen for that reason that .NET Framework provides built-in functionality for serialization and deserialization of data of this format. That allows to define serializable class which could store the app setting during its runtime. When there is a need to save data to the external file or read data from external file operations of serialization and deserialization can be made. Possible alternative to this structured file of settings can be some relational database. In the case of my project number of setting is not so great (about 50). So, it would be possible to use some simple database which is controlled by local DBMS (like SQLite). Negative moment of this approach that usage of DBMS creates unnecessary external dependencies. Because of relatively simple structure of the settings and their moderate number I've decided to use local structured file.

.NET Framework provides a feature of customizable local isolated data storage. This type of storage is a folder which path is formed by services of operating system. This path is controlled by operating system and client application can request it by using of special API. When creation of local storage is requested, the app could define desirable access permissions to the files of this storage (the storage can be set as accessible only by this app, or it can demand that any Windows account should have own version of this storage). All these services are provided automatically after the app's request. So, it is can be said that Windows local storage is some kind of light implementation of some functions of DBMS.

File of settings of my app can be accessed only from the instance of this app and any account has own version of this file. Structure of the file is shown below (figure 34).



```xml
<?xml version="1.0"?>
<SettingsData xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <MaxPower>1023</MaxPower>
  <MinPower>500</MinPower>
  <NormalPower>1000</NormalPower>
  <GPIOPwmPortNumber>18</GPIOPwmPortNumber>
  <MaxPortNumber>27</MaxPortNumber>
  <FirstRightPortNumber>17</FirstRightPortNumber>
  <SecondRightPortNumber>27</SecondRightPortNumber>
  <FirstLeftPortNumber>3</FirstLeftPortNumber>
  <SecondLeftPortNumber>4</SecondLeftPortNumber>
  <FirebaseAddress>
    https://gazeinteractor.firebaseio.com/.json
  </FirebaseAddress>
  <DefaultInterval>1800000</DefaultInterval>
  <MinInterval>1000</MinInterval>
  <MaxtInterval>20000</MaxtInterval>
  <RaspberryUser>pi</RaspberryUser>
  <RaspberryHost>192.198.0.120</RaspberryHost>
  <PictureWidth>1024</PictureWidth>
  <PictureHeight>768</PictureHeight>
  <PictureTriggerSize>100</PictureTriggerSize>
  <AttentionThresholdValue>30</AttentionThresholdValue>
  <MeditationThresholdValue>30</MeditationThresholdValue>
  <LowLevelCounterThresholdMS>10000</LowLevelCounterThresholdMS>
  <MaxMindDataValue>100</MaxMindDataValue>
  <ThinkGearPort>13854</ThinkGearPort>
  <ThinkGearIPAdress>127.0.0.1</ThinkGearIPAdress>
  <PictureUpdatingIntervalMS>20</PictureUpdatingIntervalMS>
  <TextTriggerStop>STOP\nGAZE</TextTriggerStop>
  <TextTriggerStart>START\nGAZE</TextTriggerStart>
  <PictureFontName>Tahoma</PictureFontName>
  <PictureFontSize>25</PictureFontSize>
  <TextShifht>5</TextShifht>
  <PointShift>20</PointShift>
  <TransparancyLevel>50</TransparancyLevel>
  <StartColor>Orange</StartColor>
  <StopColor>Red</StopColor>
  <RightLeftColor>Blue</RightLeftColor>
  <BackColor>Yellow</BackColor>
  <ForwardColor>Green</ForwardColor>
  <StartStopTriggerColor>White</StartStopTriggerColor>
  <BackPhraze>back</BackPhraze>
  <ForwardPhraze>forward</ForwardPhraze>
  <LeftPhraze>left</LeftPhraze>
  <RightPhraze>right</RightPhraze>
  <StopPhraze>sto p</StopPhraze>
  <PassivePhraze>passive</PassivePhraze>
  <StartPhraze />
  <TextErrorConnectionStartMessage>Failed to connect to the robot.
    Please, check your network connection and start application again.
  </TextErrorConnectionStartMessage>
  <TextErrorTitleStatrMessage>Program was stopped</TextErrorTitleStatrMessage>
</SettingsData>
```

Figure 34: Data file for string settings of the desktop (XML format)

7.6 Discussion on design of graphical user interface

Initial design of user interface was shown in the section 4.2.2.2, figure 6. Final design of user interface includes one main window. Major part of the main window is occupied by real-time video stream which is sent from the robot's camera. The app starts only in the case when there are active local network connection and Internet connection. Also, on the initial stage of launching the program checks if tis possible to establish connection with the robot (Figure 43). Video stream is shown only if there is a user in front of the computer (Figure 36). If another instance of the app is running, then the program shows message window (Figure 44) and stops its launching returning the control to the operating system.

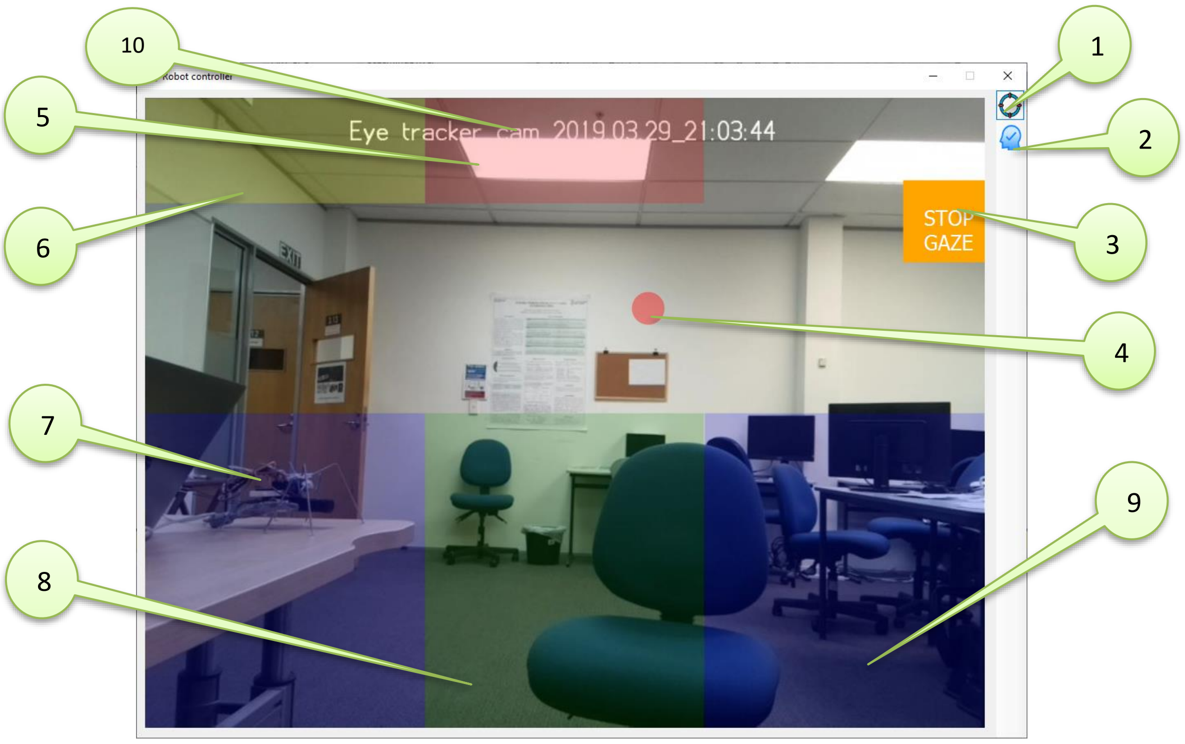


Figure 35: Main window

Table 23: Description of the user interface

1	Toggle button which enables or disables gaze input. When button is checked then areas 3, 5-9 appear (and vice versa – figure 37).	2	Toggle button which enables or disables tracking and usage of the user's brain activity during the process of driving.
3	Area which is used to start or to stop driving of robot by user's gaze. The user should gaze to this area in order to start or to stop using of gaze controlling (figure 35, 37).	4	Cursor which shows where user's gaze is directed to. Cursor can change its image in order to report current direction of robot movement. Circle means passive state of the robot (no motion).
5	Area where the user should gaze in order to stop the robot (figure 41).	6	Area where the user should gaze in order to force the robot to move backwards (figure 42).
7	Area where the user should gaze in order to force the robot to move to the left (figure 40).	8	Area where the user should gaze in order to force the robot to move forward (figure 38).
9	Area where the user should gaze in order to force the robot to move to the right (figure 39).	10	Current date and time on the robot's computer.

Main window shows real-time picture which is being captured by the robot's camera. Picture is only shown if the user is in front of the computer and mode of gaze input is on (button 1 is checked). Program forms and shows semitransparent rectangular areas which cover picture of the camera and can accept user's gazes (from the eye tracker). Each area has its own function according to the description of table 23. Functionality of the app is described in the project's requirements (Table 11).

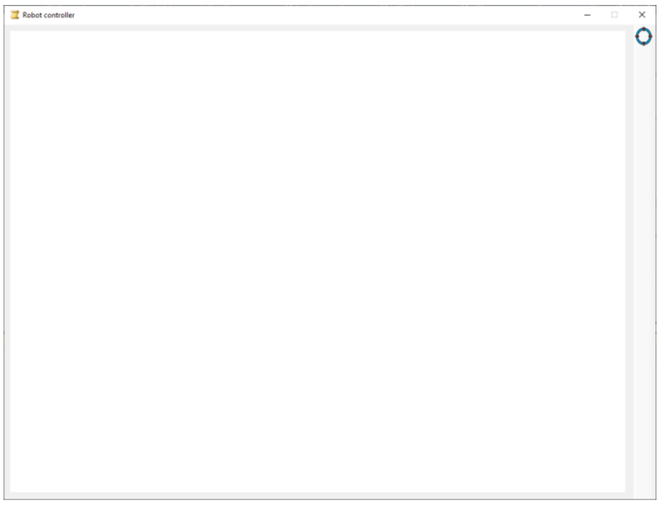


Figure 36: Main window (the user is out, or gaze controlling is off)

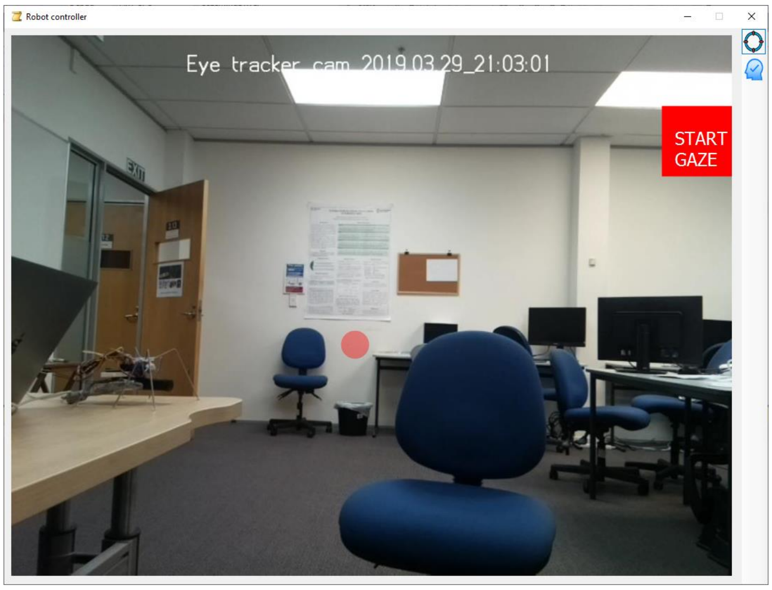


Figure 37: Mode of gaze driving is off

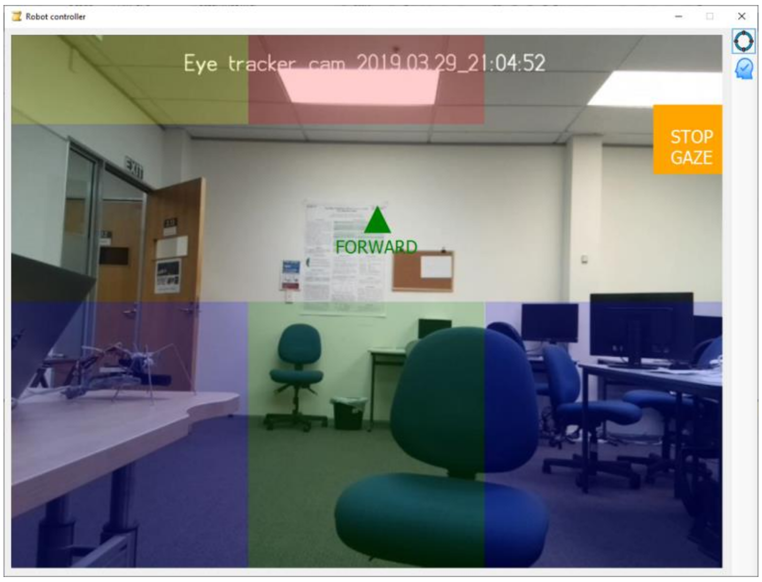


Figure 38: The robot is moving forward

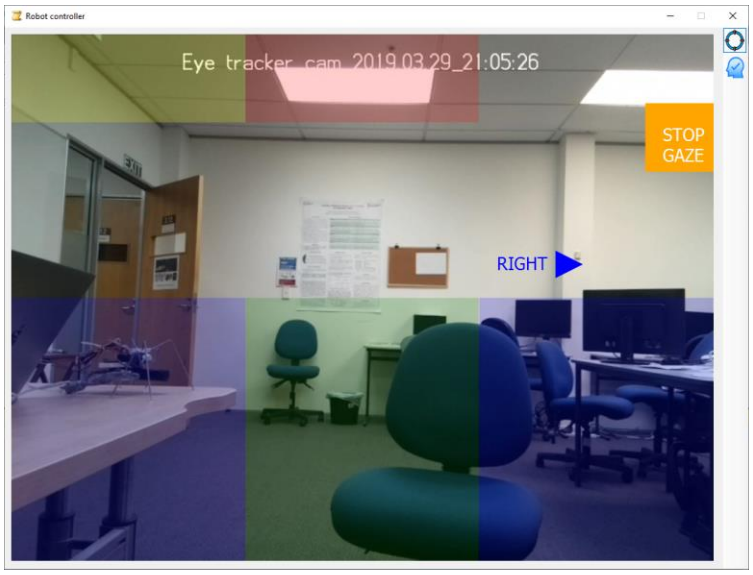


Figure 39: The robot is moving to the right

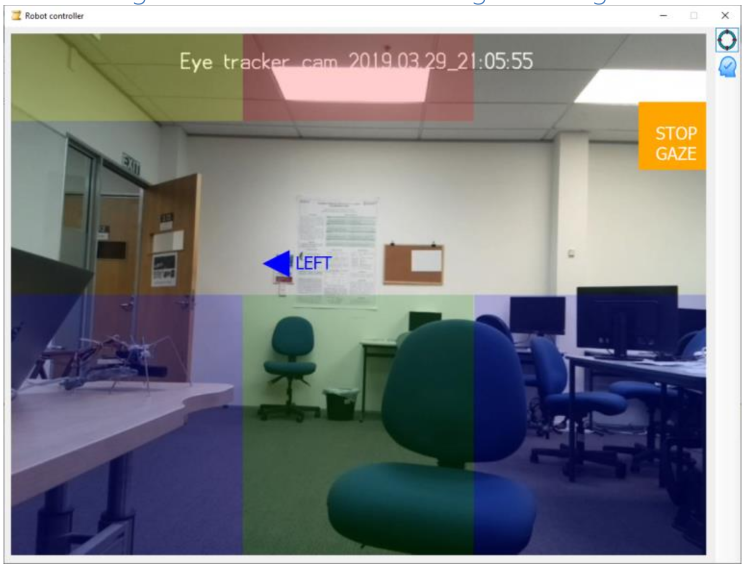


Figure 40: The robot is moving to the left

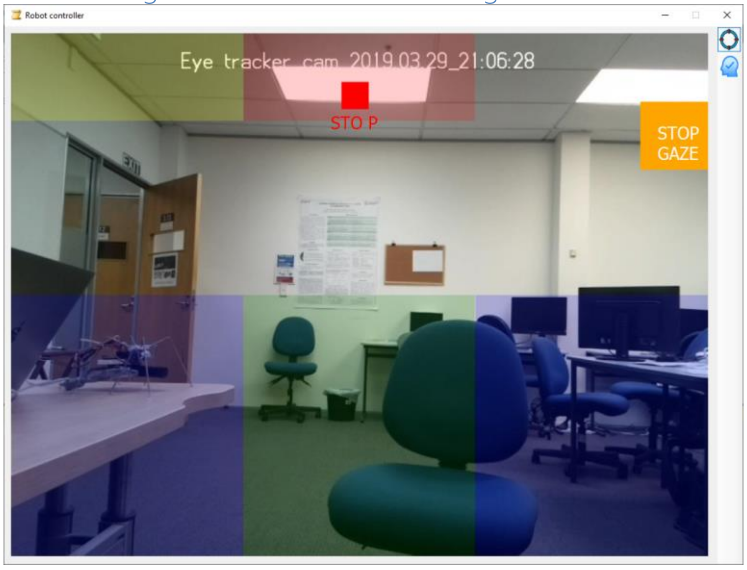


Figure 41: The robot is stopped

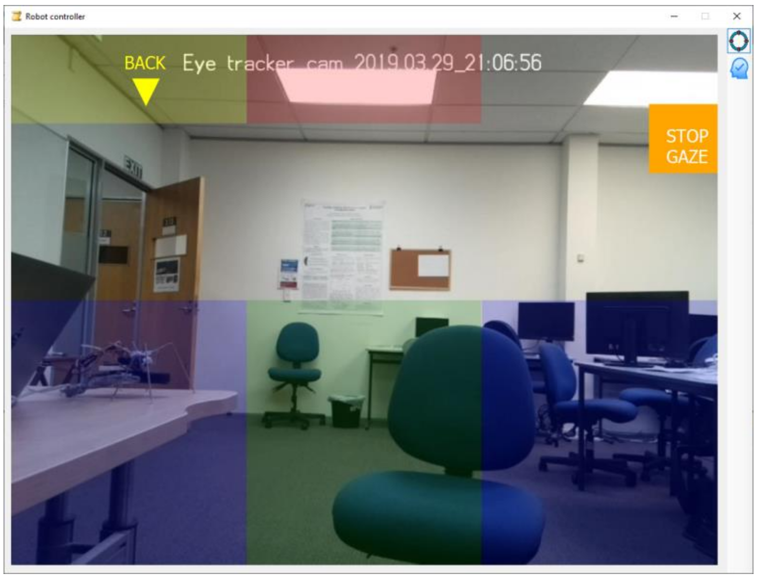
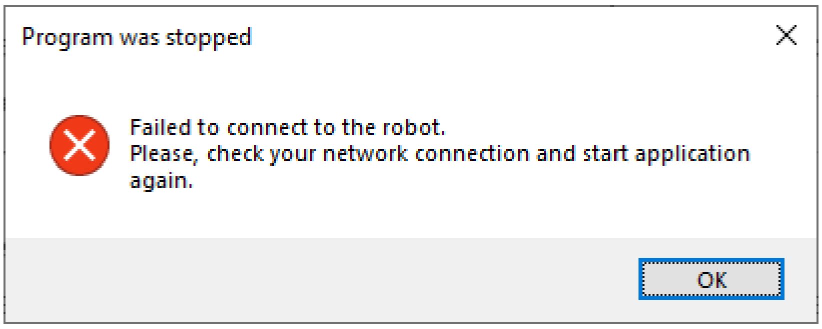


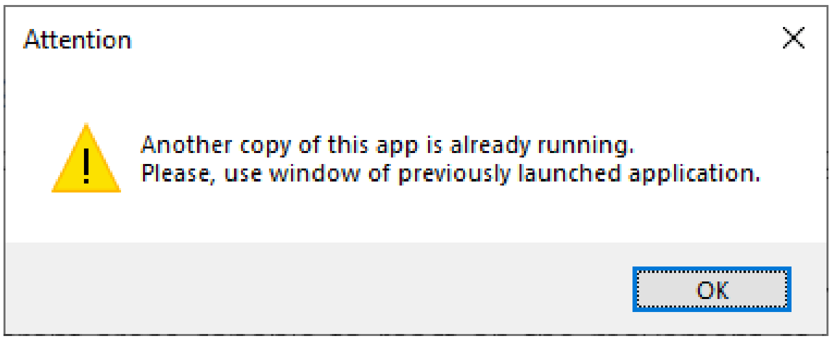
Figure 42: The robot is moving backwards


Program was stopped

Failed to connect to the robot.
Please, check your network connection and start application again.

OK

Figure 43: App's message window (impossible to establish connection)


Attention

Another copy of this app is already running.
Please, use window of previously launched application.

OK

Figure 44: App's message window (another instance of the app has been launched)

Implemented user interface reflects my understanding of possible way of usage of gaze of the user for controlling of externa entity. The main problem was to understand how and when catch user's gaze. I mean that it was clearer how to implement needed functionality technically. I decided to cover original picture with "live" transparent areas capable to react on the movement of user's gaze. But this design of UI appeared after some preliminary ideas. First idea was to crate separately located toolbar with the same functionality (its buttons would be sensitive to user's gazes). But problem of this approach is that this toolbar would be located aside video stream. It means that the user would have to detract his attention from the real situation around the robot. This distraction could lead to untimely commands and inadequate behavior of the robot. Second idea was almost the same, but I wanted to place this toolbar in front of the user's eyes on the surface of the dynamic picture. But in this case another problem arose. This toolbar would hide some fragments of the video stream what also could negatively affect the process of driving.

Final design is free of mentioned drawbacks of other earlier implementations. It shows all the picture from the camera and follows natural movements of the user's eyes. I mean that if the user looks to the left area of the window, then the robot starts moving left and so on.

This interface was tested by groupmates and was corrected according to their feedback. I changed positions of areas of backward movement (yellow) and stopping (red). Initially red area was placed in the central part just above green area (forward) and yellow area was placed just below green area. This disposition was unsuccessful and leaded to losing of adequate control of motions. Overall quality and easiness of controlling by the user's gaze was improved after corrections of positions of the mentioned areas.

Position of gaze areas and peculiarities of their behavior forced me to place main toolbar of the app on the right side of the main window (contrary to usual standard position in the top of it). That was made because right upper area is free from areas dedicated to catching of the gaze input. So, if the user needs to press a button in this area his gaze doesn't touch these areas, and this cannot cause unintended motions of the robot.

7.7 Reflection

Implementation of the software part of the system became a major part of the project. During this process I learned new architectural and technological components which allowed to achieve the needed goals. Here I would like to mention the most important and slight difficult things which were relatively new and demanded some time to understand and adapt in my software:

1. Implementation of three-layered architecture for robotic software.
   Here the main aim is to implement main functionality by using of abstractions which are decoupled from details of used hardware and localize realization of these details within one layer. This architecture influenced structure of my project but now I want to say that it would be better if my system architecture were implemented as more distributed across devices (I'm not absolutely sure but probably gaze events is better to be sent directly to the robot (such realization needs additional profiling and checking)). Performance of the robot's computer allows to implement more complicated tasks than simple sending of video stream and execution of commands. In this case it would be possible to implement more intellectual behavior of the robot (if new hardware would be added). I had idea to move some modules to the robot's side using new version of Microsoft .NET Core library which should be capable to be executed on Linux and MacOS operating systems and have the same functionality like .NET Framework (but by the moment of development this library was not yet released and was in beta status).
2. Compliance to SOLID principles of object-oriented design of software (design of system with modular structure, design and usage of interfaces, creation of classes which are responsible for rather narrow set of functions (or only one function), decoupling of classes by adapting of event-bus architecture).

Also, ideas of these principles helped to make my implementation adjustable to possible usage of new eye tracking devices. Initially it was not clear how to achieve such ability and I had to try some different ways.

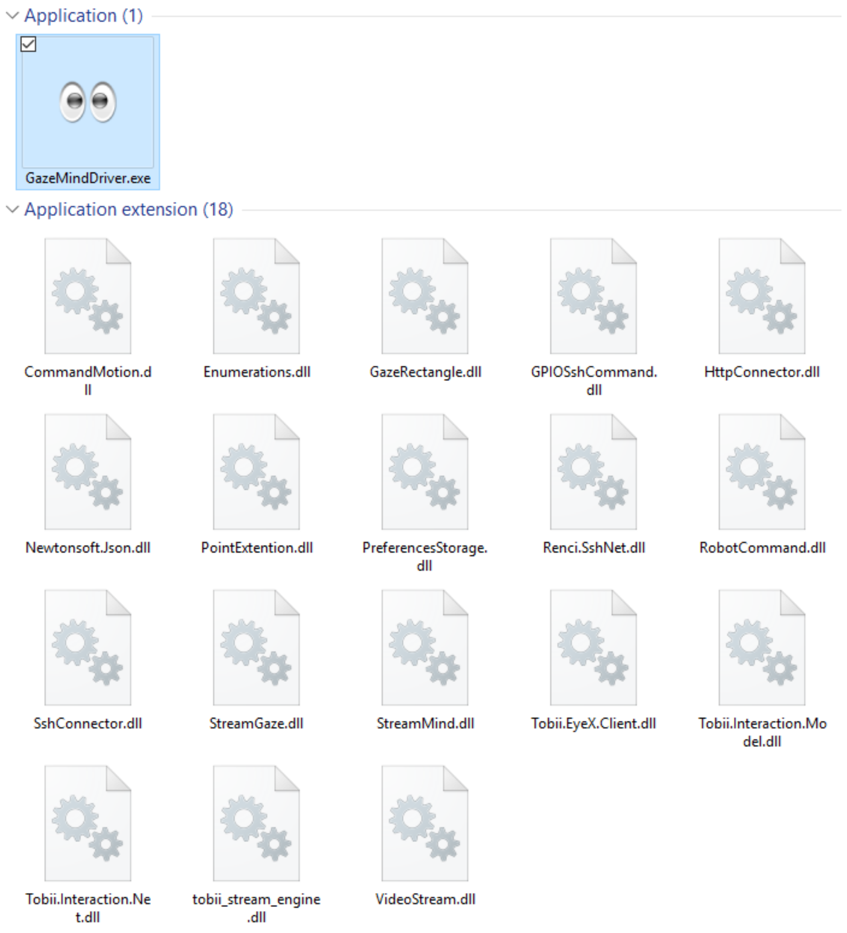
3. Creation of responsive application which processes quite large streams of data using multithreading. Here important task is to avoid problems of conflict which pertain to such style of programming (deadlocks, races and so on).
4. Usage of inter-process synchronization for launching of the only instance of the app.
5. Usage of modern features of .NET Framework and C# programming language.
6. Design of such UI that should be suitable to be controlled with user's gaze input.

At the same time, I would like to say that some functionality (not included in requirements) could be added. It can be prevention of simultaneous application's running on different computers, implementation of speed control of the robot by user's mental efforts, changing of hardware settings from graphical user interface. Also, it would be great to add some new equipment with feedback features for making the process of movement more stable and safer (for example it could be infrared sensors or GPS sensors and so on).

8 BRIEF INSTRUCTIONS ON USAGE OF THE SYSTEM

According to designed architecture is it needed to undertake some activities in order to use the system.

1. Needed equipment.
   a. Robot (see section 6.1.1)
   b. Tobi Eye Tracker 4C;
   c. NeuroSky MindWave Mobile 2 device;
   d. USB Bluetooth dongle.
2. Needed software.
   a. .NET Framework (version 4.7 or later).
   b. Desktop application "GazeMindDriver" which should be installed on the used PC (components of the app are shown below).



Application (1)

GazeMindDriver.exe

Application extension (18)

CommandMotion.dll
Enumerations.dll
GazeRectangle.dll
GPIOSshCommand.dll
HttpConnector.dll
Newtonsoft.Json.dll
PointExtention.dll
PreferencesStorage.dll
Renci.SshNet.dll
RobotCommand.dll
SshConnector.dll
StreamGaze.dll
StreamMind.dll
Tobii.EyeX.Client.dll
Tobii.Interaction.Model.dll
Tobii.Interaction.Net.dll
tobii_stream_engine.dll
VideoStream.dll

   c. NeuroSky ThinkGear Connector (on used PC).
   d. Apache Web-server installed and running on the robot's Raspberry PI computer.
   e. PHP interpreter.
   f. Python interpreter.
   a. Script RaspIPWriter.py which should start automatically with Raspberry PI operating system.

  b. Web-application http://IP_ADRESS/ /html/cam_pic.php.

Short preparation's instructions.

1. Copy folder of application "GazeMindDriver" to your computer.
2. Connect USB Bluetooth dongle and establish connection between EEG headset and Bluetooth according to instructions of EEG headset.
3. Connect Tobii Eye Tracker 4C according to manual of this device.

Usage instructions.

1. Turn on the robot.
2. Turn on EEG headset and take it on.
3. Launch the mobile application "GazeMindDriver".
4. Follow instructions on the screen of the app.

9 TESTING

9.1 Testing plan

Table 24: Testing plan for the project

Code	Name of testing	Expecting result
T-P	Performance testing.	The app should comply requirements PR-1 - PR-2.
T-F	Functional testing.	The app should comply requirements SFR-1 - SFR-11.

Table 25: Performance testing plan

Test case code	Tested requirement	Expected result
T-PR1	PR-1	The application should provide the user with the latest picture from the robot's camera with possible latency not more than 1 second (under the condition that appropriate quality LAN connection is provided).
T-PR2a	PR-2a	System should guaranty immediate reaction of UI on gaze events (within defined temporal scope): if fixation inside of some GCR is reported, then command should be sent to the robot immediately (not longer than 1 sec.).
T-PR2b	PR-2b	System should guaranty immediate reaction of UI on gaze events (within defined temporal scope): if place of fixation of gaze is changed, then current visible graphical representation (special cursor) of the robot movement must be changed immediately (not longer than 1 sec.).

Table 26: Functional testing plan

Test case code	Tested requirement	Expected result
T-F1	SFR-1	Only one instance of running application is allowed for current Windows account.
T-F2a	SFR-2a	The app should check the state of network connection during the time of its functioning. a. if connection is ok, then commands to the robot can be sent.
T-F2b	SFR-2b	The app should check the state of network connection during the time of its functioning. b. if connection is broken, then control commands are blocked and not sent.
T-F3	SFR-3	The application should store characteristics of its hardware and software components in external isolated data storage provided by Windows operating system in the form of XML file.
T-F4	SFR-4	Each Windows user's account should have its own copy of the app's data storage file.
T-F5a	SFR-5a	User's gaze at trigger GCR should turn alter the mode of control of the robot movements. a. If gaze control is on, then trigger GSR should turn it off.
T-F5b	SFR-5b	User's gaze at trigger GCR should turn alter the mode of control of the robot movements.

		b. If gaze control is off, then trigger GSR should turn it on.
T-F6	SFR-6	User's gaze at "Forward GCR" should force the robot to start moving to the direction where its camera is directed (forward moving).
T-F7	SFR-7	User's gaze at "Backward GCR" should force the robot to start moving to the direction which is opposite to where its camera is directed (backward moving).
T-F8	SFR-8	User's gaze at "Turn Left GCR" should force the robot to start moving left relative to the direction in which its camera is directed.
T-F9	SFR-9	User's gaze at "Turn Right GCR" should force the robot to start moving right relative to the direction in which its camera is directed.
T-F10	SFR-10	User's gaze at "Stop GCR" should force the robot to stop (if it's moving).
T-F11a	SFR-11a	The application should provide the user with ability to use EEG headset for controlling of movement of the robot: a. if level of overall brain activity reported by EEG headset is lower than predefined value, then movement of the robot should be stopped, and control of the robot by the user's gaze should be prevented;
T-F11b	SFR-11b	The application should provide the user with ability to use EEG headset for controlling of movement of the robot: b. if level of overall brain activity reported by EEG headset is equal or higher than predefined value, then movement of the robot should be allowed.

9.2 Performance test case T-PR1

The goal of this test case is to guarantee that the app must show the latest picture from the robot's camera. This is important because only actual picture allows to control robot in on-line mode. Possible latency could make gaze control irrelevant to current position of the robot because the picture shows past, but not current situation. According to requirements and common sense, possible latency must be not more than 1 second (the less the better).

Latency of the video stream is inevitable and depends on several factors. In my project these factors are speed of local network connection, algorithms of processing of video stream (and overall efficiency of the app and a balance between this task and other tasks which are executing simultaneously), and efficiency of robot-side video streaming service.

First factor is out of control of my system. I measured speed of wireless local network connection which is provided by EDENZ (I used Meteor software (OpenSignal, 2018)). Average speed of uploading and downloading of Wi-Fi connection on the 3rd level of EDENZ building is about 30-40Mbs. Wireless Wi-Fi connection is used to implement interaction between the robot and controlling PC (the robot is mobile device and can use Wi-Fi connection only).

Usage of some specific software and hardware also affects efficiency of showing if video streaming. Hardware part (video camera) was provided as it is (be the college) and it is not a question of discussion. Software part of the process relies on built-in robot's Raspberry PI computer's "*raspivid*" utility. Any video service for Raspberry PI uses this utility somehow for processing of video capturing from standard Raspberry PI camera. Here I would say that even in local (non-network) mode built-in command sometimes demonstrated some latencies. My project is not about configuring hardware and low-level software development, but I would say that there is some room for optimization of software components of Raspberry PI computer. Anyway, my experiments and searches showed that there are rather narrow filed of choice of software components which can be used to deliver low-latency network video streaming from Raspberry PI. In 3 cases of 4 (I tried 4 different services for network video streaming) latency of video stream which was received over local network connection was so significant that was easily detected visually (some video streaming software had delays up to 3 to 6 seconds!). So, I would say that first stage of performance testing was quite informal because there was no need to measure of that which was simply visible. Final choice of video streaming software component was made on the basis of minimal time of delay. The best result was provided by RPi_Cam_Web_Interface (Elinux.org, 2019) which relies on Apache-web server and "*raspivid*" utility.

Another factor which affect efficiency of video output of the app is overall balance of simultaneous processes that are being executed during runtime. Section 7.1.1 was devoted to discussion about ways of achieving of optimal performance by using of multithreading. This part is under my control (to a certain extent, of course). Introducing and implementation of this approach allowed to significantly reduced latencies of video streaming during runtime. That could be noticed even visually, but I undertook performance testing (Molyneaux , 2015) in order to prove this fact more formally and eliminate subjective personal judgements.

Method of this test is based on the measurements of difference between two moments of time. First moment is when picture was sent from the Raspberry PI. Second moment of time is when received picture was shown in the video frame of the main window. Shown picture appeared after preliminary graphical processing and contained not only original video output but also additional elements which were drawn on its surface. These elements are the gaze cursor and 6 rectangular areas which can catch user's gazes. The first moment is reported by Web-app which sends a picture to the client (my app) by its request. The second moment is reported by the method which is responsible for drawing of the cursor and rectangles. Following table shows results of measurements of different stages of the project. I show measurements before and after I started to use multithreaded processing in my app.

Each test is a measurement session which consists of series of 100 consecutive measurements (each session gives one average value of time span in seconds). All tests used the same Raspberry PI service for video streaming.

Speed of Wi-Fi connection is measured once before each cycle of tests. One particular step of the measurement process cycle is done as follows. The server side takes picture from the camera and sends it to the app with attached data (picture id, creation time). Then the app receives data, processes picture (adds gaze rectangles and gaze cursor), shows picture in the main window and then calculate length of time which passed from the moment of sending to moment of showing.

Table 27: Test case T-PR1 (results for non-multithreaded implementation)

Measurement session	Average speed of local network connection during the session (downloading in Mbps) in room 3.13	Average time span (in seconds) when robot was inside room 3.13	Test result
1	36.25	3.02	Failed
2	40.8	2.44	Failed
3	34.8	3.21	Failed
4	48.5	1.02	Passed
5	38.2	4.07	Failed
Average value	39.71	2.75	Failed (average time should me less than 1).

The next test shows measurements which were undertaken after introduction of multithreaded architecture, when I implemented threads (by usage of .NET thread pool) for following tasks:

1. gaze tracking and processing,
2. brain data tracking and processing,
3. getting of the picture from the robot over network connection,
4. sending of each motion command inside its own thread.

Table 28: Test case T-PR1 (results for multithreaded implementation)

Measurement session	Average speed of local network connection during the session (downloading in Mbps) in room 3.13	Average length of time span (in seconds) when robot was inside room 3.13	Test result
1	34.8	1.20	Failed
2	42.4	0.64	Passed
3	36.7	0.87	Passed
4	42.6	0.57	Passed
5	36.4	0.91	Passed
Average value	38.26	0.84	Passed (average time is less than 1).

Result of these tests shows that undertaken activities allowed to reduce latency in video streaming. But anyway, there is an unpredictable factor such as speed of Wi-Fi connection. The most important result for me is that my part of job positively influenced overall performance and my efforts on implementing multithreading architecture was not undertaken in vain.

Also, I probably need to somehow defend the way I implemented these tests. I measured needed lengths of time spans inside my app (by invocation of methods for processing of moments of time and periods of time) and didn't use external tools. This way was chosen because standard performance measurement tools (Molyneaux , 2015) like JMeter and others demand representation of my data in some specific third-party format and they are not ideally suitable for my task (because the task involves several hardware components). If I used standard tools, I would have to create additional algorithms to adapt my program to these tools. I believe that usage of extraneous software is not expedient in this case.

9.3 Performance test cases T-PR2a and T-PR2b

Table 29: Test cases T-PR2a,b

T-PR2a	PR-2a	System should guaranty immediate reaction of UI on gaze events (within defined temporal scope): if fixation inside of some GCR is reported, then command should be sent to the robot immediately (time of sending is no longer than 1 sec.).
T-PR2b	PR-2b	System should guaranty immediate reaction of the robot on gaze events (within defined temporal scope): the robot should start executing of received command not later than 1 second from the moment when this command was sent.

This group of requirements also pertain to performance. If these requirements are met, then the system behaviour should be considered as responsive. As was mentioned in the section 5.2, timespan length of immediate reaction is based on researches (Nielsen, 1993) of peculiarities of human subjective perception of time frames of system's UI responsiveness. Also, PR-2b sounds more than naturally because immediate reaction of the robot on given commands is expected by users (if only the robot is running not on the Moon). Negative factors for these aspects of the system are necessity of parallel processing of the video and possible delays because of poor network connections. Technique of measurements for this part of testing is almost the same as described in 9.2.

Table 30: Test case T-PR2a

Measurement session	Average length of time span (in seconds) between gaze fixation and sending of command (100 consecutive measurements)	Test result
Before usage of multithreading	0.47	Passed
After usage of multithreading	0.06	Passed

The final result of measurements shows that length of time span after implementation of multithreading is one order of magnitude shorter than initial.

Table 31: Test case T-PR1 (results for multithreaded implementation)

Measurement session	Average length of time span (in seconds) between gaze command and starting of its execution (50 consecutive measurements)	Test result
Before usage of multithreading	0.5	Passed
After usage of multithreading	0.1	Passed

Concerning the second result, I would say that not only the time span was shortened but also UI responsiveness became better. Each command is sent from its own thread which allows to improve overall performance of the system.

9.4 Functional test cases

Table 32: Test case T-PR1 (results for multithreaded implementation)

Test case code	Tested requirement	Expected result	Result of testing
T-F1	SFR-1	Only one instance of running application is allowed for current Windows account.	Passed
T-F2a	SFR-2a	The app should check the state of network connection during the time of its functioning. a. if connection is ok, then commands to the robot can be sent.	Passed
T-F2b	SFR-2b	The app should check the state of network connection during the time of its functioning. b. if connection is broken, then control commands are blocked and not sent.	Passed
T-F3	SFR-3	The application should store characteristics of its hardware and software components in external isolated data storage provided by Windows operating system in the form of XML file.	Passed
T-F4	SFR-4	Each Windows user's account should have its own copy of the app's data storage file.	Passed
T-F5a	SFR-5a	User's gaze at trigger GCR should turn alter the mode of control of the robot movements. a. If gaze control is on, then trigger GSR should turn it off.	Passed
T-F5b	SFR-5b	User's gaze at trigger GCR should turn alter the mode of control of the robot movements. b. If gaze control is off, then trigger GSR should turn it on.	Passed
T-F6	SFR-6	User's gaze at "Forward GCR" should force the robot to start moving to the direction where its camera is directed (forward moving).	Passed

T-F7	SFR-7	User's gaze at "Backward GCR" should force the robot to start moving to the direction which is opposite to where its camera is directed (backward moving).	Passed
T-F8	SFR-8	User's gaze at "Turn Left GCR" should force the robot to start moving left relative to the direction in which its camera is directed.	Passed
T-F9	SFR-9	User's gaze at "Turn Right GCR" should force the robot to start moving right relative to the direction in which its camera is directed.	Passed
T-F10	SFR-10	User's gaze at "Stop GCR" should force the robot to stop (if it's moving).	Passed
T-F11a	SFR-11a	The application should provide the user with ability to use EEG headset for controlling of movement of the robot: 1. if level of overall brain activity reported by EEG headset is lower than predefined value, then movement of the robot should be stopped, and control of the robot by the user's gaze should be prevented;	Passed
T-F11b	SFR-11b	The application should provide the user with ability to use EEG headset for controlling of movement of the robot: 2. if level of overall brain activity reported by EEG headset is equal or higher than predefined value, then movement of the robot should be allowed.	Passed

9.4 Reflection on testing

Testing process of this project was implemented by following the planned quality assurance process (3.7) and requirements. Not all activities of the testing process were described in this report. Here I didn't mention about units testing and acceptance testing. My groupmates took part in acceptance testing and gave me some valuable feedback. Also, I was trying to create and run unit tests as soon new methods were firstly introduced (or even before they were introduced). This approach is called test driven development (Bender & McWherter, 2011). I cannot state that I followed this approach 100% percent of time, but I did my best to develop quality software. I used .NET built-in testing tools and third-party xUnit library. Quality of the source code was checked by JetBrains ReSharper (third-party component of Visual Studio).

Besides that, I used profiling of Visual Studio development environment. While process of debugging of running application, this tool allows to measure level of usage of operational memory and processor time. This tool made it possible to find weaknesses in the algorithms of my program's and to minimize amount of resources being consumed.

The most interesting and challenging task was to realize measurement of the system performance in order to guaranty quite quick reaction of the system on the user's actions. I would say that testing showed that usage of Wi-Fi connection was not correct decision. Despite the fact that I used equipment provided by the college, it is important to notice that casting of video form the robot should be implemented by more reliable channel (also, negative effect probably was related with rather slow speed of connection and not efficient enough processing of video by built-in Raspberry PI software components).

Because of lack of time I didn't pay enough attention to integration testing and non-functional testing. Testing of interaction between components of my system is important task, because I created 16 DLL files and 1 EXE file of executable modules. And, yes, all of these modules interact. I would say that integration testing was implemented not by strict plan but more intuitively on the base of visible effect of software functioning (and results of performance tests and profiling). In future project it would be great to pay more attention to this type of tests. The same could be said about non-functional testing. My approach in this case also was based on previous expertise and intuitive understanding of things which is not good in strict sense of engineer science.

10 Conclusion

10.1 Reflection

Overall experience I gained from this project can be considered as positive. I created such type of project which I always wanted to implement. Creation of system which includes standard personal computer and external mechanical system which can be controlled by such unusual interfaces as gaze input and BCI is really challenging and fascinating task. During the process of design and development I learned not only technological components of such systems but also learned a lot of new ideas from the field of software engineering.

If to say about difficult or unsuccessful moments of this capstone I need to mention the following:

1. That was not very clear from the beginning what kind of software architecture should be implemented in the project. Firstly, I spent some time for exploring some architectural patterns which are more suitable for desktop or web-systems (web-sites). They were not appropriate because do not involve interaction with hardware and assume that any computer application is not more than a client of some database. I found that real projects of robotic software systems use different variations of multi-tiered architectural patterns and after that I adapted one of them.
2. I didn't manage to implement complete and systematic test process because of lack of experience in the projects like this (which includes interacting hardware and software components). This project showed that I need to plan such activities better before starting the development.
3. Also, I would like to have more efficient process of video processing in my program. This problem could be solved by usage of other hardware and other channels of data transferring. And this could demand from me learning of new ways of integration of these things to my project.

The most interesting positive results for me in this project are:

1. Learning of new ways of interaction between users and computers (eyes movements detection and brain-computer interface),
2. Implementation of these interfaces in my own software system,
3. Learning of new APIs and SDKs and their practical usage
4. New additional experience of analysis of ideas of software design and architecture and their practical usage in my project,
5. New experience in testing and implementation of quality characteristics,
6. Planning and assembling of the robot, analysis and choice of hardware and software components of the robot,
7. Development of software for robot controlling,
8. Development of complex software and hardware system which contains of set of interacting modules,
9. Searching of the ways of implementing stable system with reliable level of performance,

10. Implementation of multithreading architecture of the app,
11. Working with modern version of .NET Framework library and C# programming language,
12. Working with modern development tools such as Microsoft Visual Studio Community 2017 and JetBrains ReSharper, xUnit testing API.

But there is not any perfection in the world. Possible enhancements and improvements of the project are listed below.

Improvements in hardware part of the project:

• Usage of more sophisticated moveable part which capable to report about its current state. Such an equipment can deliver higher level of responsiveness of both hardware and software.

• Usage of additional sensors. One of the weak moments of the implemented functionality is that the robot can collide with obstacles when user fails to react in time by sending command by one's gaze. Additional sensors could make process of driving more secure and the system would be considered as more reliable.

• Usage of more advanced video camera. New camera should provide more stable video stream. Also, there is new generation of such devices which allow to show view of 360 degrees around controlled device. In the case usage of this type of camera the user can control backward movements and see actual picture behind the robot. But this can make the project more expensive.

Software part also can be improved as follows.

• Current implementation of the system uses Windows desktop application as a receiver of user’s gazes. Ideally, it would be better to make this part more universal (not so tightly dependent form Windows operating system). This could be done by introducing of microservice architecture and usage of newest (not yet released) versions of .NET Core library (coming version is going to allow launching standard Windows Forms applications in Linux and MacOS operating systems). Microservice could play the role of a supplier of data from gaze devices. It also should be capable to work with gaze devices of different types.

• It is needed to implement more stable and efficient algorithms for providing of smoother video rendering.

• Another possible modification of the project is connected to understanding of brain activities during the process of controlling of the device by gaze input. It would be useful to learn more precisely about patterns of brain activity of a driver (during the process of driving) because this can make controlling of the robot more reliable. Testing process shows that connection between levels of attention and relaxation and ability to control the robot is not so obvious. In my current implementation I stop motion of the robot when level of attention and relaxation drop below predefined value.

References

Admiraal, H. (2019, March 8). *Pitfalls using UML in RUP*. Retrieved from Pitfalls using UML in RUP: https://sparxsystems.com/downloads/whitepapers/Pitfalls%20using%20UML%20in%20RUP%20_part%202_.pdf

Ahmad, A., & Ali Babar, M. (2016). Software Architectures for Robotic Systems: A Systematic Mapping. *The Journal of Systems and Software*, 16-39.

Ajzele, B. (2017). *Mastering PHP 7*. Birmingham: Packt Publishing.

Arnuphaptrairong, T. (2011). Top Ten Lists of Software Project Risks:Evidence from the Literature Survey. *Proceedings of the International MultiConference of Engineers and Computer Scientists*. Hong Kong: IMECS 2011.

Aspray, W. (1990). *John von Neumann and the Origins of Modern Computing*. Cambridge: The MIT Press.

Barnes, J. (2007). *Implementing the IBM Rational Unified Process and Solutions*. New Jersey: IBM Press.

Bender, J., & McWherter, J. (2011). *Professional Test Driven Development with C#*. Hoboken: Worx Press.

Bijholt, A., Femke, N., Gerritsen, S., Poel, M., & Braam, L. (2015). Usability of Three Electroencephalogram Headsets for Brain–Computer Interfaces: A Within Subject Comparison. *Interacting with Computers*, 500-511.

Bitlner, I., & Spence, I. (2006). *Managing Iterative Software Development Projects*. Boston: Addison-Wesley Professional.

Boggs, W., & Boggs, M. (2002). *Mastering UML with Rational Rose 2002*. Alameda: Sybex.

Booch, G. (1991). *Object oriented design with applications*. Redwood: The Benjamin/Cummings Publishing Company, Inc.

Cervantes, H., & Kazman, R. (2016). *Designing Software Architectures*. Pearson Education.

Cleary, S. (2014). *Concurrency in C# Cookbook: Asynchronous, Parallel, and Multithreaded Programming*. O'Reilly Media.

Clements, A. (2006). *Principles of Computer Hardware*. Oxford: Oxford University Press.

Drogon. (2019, January 23). *Wiring Pi*. Retrieved from Wiring Pi - GPIO Interface library for the Raspberry Pi: http://wiringpi.com/the-gpio-utility/

Dwivedi , H. (2003). *Implementing SSH: Strategies for Optimizing the Secure Shell*. San Francisco: Wiley.

Elinux.org. (2019, January 23). *RPi-Cam-Web-Interface*. Retrieved from Embedded Linux Wiki: https://elinux.org/RPi-Cam-Web-Interface

Emotiv Inc. (2018, October 23). *Emotiv - Leaders in wireless EEG brain monitoring technology*. Retrieved from Homepage - Emotiv: https://www.emotiv.com/

Flowler, M. (2013). *Patterns of Enterprise Software Architecture*. Addison-Wesley.

Gamma, E., Helm, R., Jonson, R., & Vlissides, J. (1994). *Design Patterns: Elements of Reusable Object-Oriented Software*. New York: Addison-Wesley Professional.

Gomaa, H. (2000). *Designing Concurrent, Distributed, and Real-time Applications With UML*. Addison-Wesley.

Hunt, A., & Tomas, D. (2000). *The pragmatic programmer*. Boston: Addison-Wesley.

IBM Corp. (2019, March 8). *Rational Unified Process*. Retrieved from RUP Homepage: https://sce.uhcl.edu/helm/rationalunifiedprocess/

IncludeHelp. (2019, January 21). *L293D Motor Driver IC*. Retrieved from IncludeHelp.com: https://www.includehelp.com/embedded-system/l293d-motor-driver-ic.aspx

InteraXon Inc. (2018, October 23). *MUSE ™ | Meditation Made Easy*. Retrieved from MUSE ™ | Meditation Made Easy: http://www.choosemuse.com/

Kortenkamp, D. (2015). *Handbook of Robotics*. Pittsburgh: Robotics Institute. Retrieved from Handbook of Robotics: https://pdfs.semanticscholar.org/6b97/871bffa7577a26fc5a6b1c1edab33c7a07b7.pdf

Kumar, D., & Arjunan, S. (2016). *Human-Computer Interface Technologies for the Motor Impaired*. CRC Press.

Langer, A. (2016). *Guide to Software Development. Designing and Managing the Life Cycle*. New York: Springer.

Larman, C. (2004). *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development*. Boston: Addison Wesley Professional.

Leigh, J. (2015). *The Neurology of Eye Movements*. Oxford: Oxford University Press.

Mallawaarachchi, V. (2018, August 21). *What is an Architectural Pattern?* Retrieved from Medium – a place to read and write big ideas and important stories: https://towardsdatascience.com/10-common-software-architectural-patterns-in-a-nutshell-a0b47a1e9013

McLean, G. (2014). *Adaptive Code via C#: Class and Interface Design, Design Patterns, and SOLID Principles*. Redmond: Microsoft Press.

Microsoft Corp. (2019, February 18). *Gaze interactions and eye tracking in UWP apps*. Retrieved from Windows Dev Center: https://docs.microsoft.com/en-us/windows/uwp/design/input/gaze-interactions

Microsoft Corporation. (2017). *Modernize existing .NET applications with Azure cloud and Windows Containers*. Redmond: Microsoft Press and Microsoft DevDiv.

Microsoft Corporation. (2018, April 5). *Design basics for Desktop applications*. Retrieved from Microsoft Docs: https://docs.microsoft.com/en-us/windows/desktop/uxguide/designprinciples

Molyneaux , I. (2015). *The Art of Application Performance Testing: From Strategy to Tools*. Newton: O'Reilly Media.

Muhlestein, R. (2018, September 1). *Operator Overloading is Evil*. Retrieved from Medium: https://medium.com/@robmuh/operator-overloading-is-evil-8052a8ae6c3a

NeuroSky Inc. (2018, October 25). *NeuroSky Developer Tools*. Retrieved from NeuroSky Developer - Docs: http://developer.neurosky.com/docs/doku.php?id=developer_tools

NeuroSky Inc. (2018, October 26). *Useful, Applicable Brain Activity Algorithms*. Retrieved from NeuroSky Inc. Body and Mind. Quantified.: http://neurosky.com/biosensors/eeg-sensor/algorithms/

NeuroSky, Inc. (2018, October 23). *EEG - ECG - Biosensors*. Retrieved from NeuroSky | Body and Mind. Quantified.: http://neurosky.com/

NewroSky Inc. (2018, October 25). *ThinkGear Serial Stream Guide*. Retrieved from NeuroSky Developer Program: http://developer.neurosky.com/docs/doku.php?id=thinkgear_communications_protocol

Nielsen, J. (1993). *Usability Engineering.* Morgan Kaufmann.

Nixon, D. (2015). *Getting Started with Python and Raspberry Pi.* Birmingham: Packt Publishing.

OpenSignal. (2018, December 11). *Meteor iOS and Android from OpenSignal*. Retrieved from Meteor iOS and Android from OpenSignal: https://meteor.opensignal.com/

Peicevic, A. (2016). *Apache HTTP Server introduction.* CreateSpace Independent Publishing Platform.

Perry, G. (1992). *Moving from C to C++.* Midwest City: Sams publishing.

Pressman, R. (2001). *Software engineering.* New York: McGraw-Hill.

Raspberry Pi. (2016). *Documentation*. Retrieved from https://www.raspberrypi.org/documentation/

Rational company. (2001). *Rational Unified Process: Best Practices for Software Development Teams.* Rational the Software Development Company.

ResearchGate. (2019, March 13). *The Elaboration and the Construction phases of the RUP*. Retrieved from ResearchGate: https://www.researchgate.net/figure/The-Elaboration-and-the-Construction-phases-of-the-RUP-were-unified-and-form-the-PROLES_fig1_320491013

Sheriff, N., & Lazar, G. (2018). *End to End GUI Development with Qt5.* Bermigham-Mumbai: Packt.

Silikoid. (2019, January 22). *Simple Motor Board for the Raspberry Pi with IC L293d and software pulse-width modulation*. Retrieved from Knight of PI: http://www.knight-of-pi.org/simple-dc-motor-board-for-the-raspberry-pi-with-ic-l293-and-software-pulse-width-modulation/

Stoneburner, G., Goguen, A., & Feringa, A. (2002). *Risk Management Guide for Information Technology Systems*. Gaithersburg: National Institute of Standards and Technology.

Stroustrup, B. (2013). *The C++ Programming Language, 4th Edition.* Addison-Wesley Professional.

Swanson, D. (2016). *Software Quality Assurance: Integrating Testing, Security, and Audit.* Boca Raton: Auerbach Publications.

Thalmic Labs. (2018, October 10). *Myo Gesture Control Armband*. Retrieved from Myo Gesture Control Armband: https://www.myo.com/

The Qt Company. (2019, March 9). *Qt Documenation*. Retrieved from Development Resources: https://www.qt.io/developers/

The Tobii Group. (2018, October 10). *Tobii.com - Tobii is the world leader in eye tracking*. Retrieved from Tobii Developer Zone - Eye Tracking SDK downloads, documentation and forum: https://developer.tobii.com/

Tobii Group. (2018, December 10). *Tobii API Reference*. Retrieved from Tobii Core SDK: https://tobii.github.io/CoreSDK/api/index.html

Trent, S., Tatsubori, M., & Suzuruma, T. (2019, February 5). *Performance Comparison of PHP and JSP as Server-Side Scripting Languages*. Retrieved from ResearchGate: https://www.researchgate.net/publication/225161349_Performance_Comparison_of_PHP_and_JSP_as_Server-Side_Scripting_Languages

Troelsen, A. (2017). *Pro C# 7: With .NET and .NET Core*. Minneapolis: Apress.

Vance, S. (2013). *Quality Code: Software Testing Principles, Practices, and Patterns*. Boston: Addison-Wasley.

Villela, R. (2019). *Pro .NET Framework with the Base Class Library*. Minneapolis: Apress.

Wallace, L., & Keil, M. (2004). Software Project Risks and their Effect on Outcomes. *Communication fo the ACM*, 68-73.

Warren, G. (2018). *Advanced Raspberry Pi: Raspbian Linux and GPIO Integration*. New York: Apress.

Yemul, R. S. (2015). *Procedure Oriented Programming Using C*. Nirali Prakashan.

Zaphiris, P., & Ang, C. (2008). *Human Computer Interaction. Concepts, Methodologies, Tools, and Applications*. Citeseer.